Applied Business Analytics for Marketing Decision-Making

Business Analytics and Data Visualization

Dr. Jose Mendoza

Academic Director and Clinical Associate Professor

MS in Integrated Marketing

New York University

School of Professional Studies

Fall 2026

Copyright

Applied Business Analytics for Marketing Decision-Making: Business Analytics and Data Visualization

Copyright © 2026 by Jose Mendoza. All rights reserved.

This course material and reading guide was prepared for instructional use in Business Analytics and Data Visualization (INTG1-GC 2300) in the MS in Integrated Marketing program at the New York University School of Professional Studies.

This guide is distributed to enrolled students in chapter format through the course learning management system. Students may use the assigned chapters, datasets, examples, templates, and exercises for their own learning in this course. No part of this work may be reproduced, distributed, sold, posted publicly, or transmitted outside the course without prior written permission from the author, except for brief quotations used for academic purposes with appropriate citation.

Suggested citation: Mendoza, J. (2026). *Applied business analytics for marketing decision-making: Business analytics and data visualization*. New York University School of Professional Studies.

Product and company names mentioned in this guide (including ChatGPT, Claude, Gemini, GitHub Copilot, Google Colab, NotebookLM, and Tableau) are trademarks of their respective owners and are used for identification purposes only. Their mention does not imply endorsement.

Dedication

To my students, whose questions continue to show that analytics is not only about finding answers, but about learning how to ask better questions.

Preface

Business analytics is changing.

For many years, learning analytics meant learning to perform a sequence of technical tasks: clean a dataset, run a model, create a chart, and interpret the output. Those tasks still matter. Students still need to understand data, measurement, relationships, models, visualizations, and recommendations. But the way analysts perform this work has changed.

Today, business analysts increasingly work with AI assistants. Tools such as ChatGPT, Claude, Gemini, NotebookLM, GitHub Copilot, Colab AI features, Tableau AI capabilities, and other emerging platforms can generate code, explain errors, summarize data, suggest analyses, draft interpretations, and create first versions of charts and dashboards. These tools are becoming part of everyday professional work.

But faster output does not automatically produce better analysis.

AI tools can be useful, but they can also be wrong, vague, overconfident, or misleading. They may produce code that runs but answers the wrong question. They may generate charts that look polished but distort the message. They may summarize findings in language that sounds convincing but goes beyond what the data supports. In an AI-assisted environment, the analyst's responsibility does not disappear. It becomes more important.

This guide is written for that environment.

The goal is not to turn every student into a software engineer or statistician. The goal is to help future marketing and business professionals become capable, critical, and responsible users of analytics. Students will learn how to prepare data, explore relationships, build preliminary models, evaluate evidence, create visualizations, and communicate recommendations. They will also learn how to work with AI tools without surrendering judgment to them.

Modern marketing analysts do not merely produce code, charts, models, or dashboards. They frame problems, supervise AI-assisted workflows, verify outputs, and translate evidence into decisions.

Throughout this guide, students will use AI assistants as part of the analytics process. But AI use is not treated as a shortcut. It is treated as a professional workflow that requires discipline. Students must specify what they are trying to do, predict what should happen, verify what was produced, explain the result, and document the process.

Analytics becomes valuable only when it improves decision-making. That principle guides every chapter.

Who This Guide Is For

This guide is designed for graduate students in marketing, integrated marketing, business analytics, and related professional programs. It assumes that students may have limited prior experience with programming or statistics, but that they are preparing to make or support decisions in data-rich business environments.

The guide is especially appropriate for students who need to:

- Use data to support marketing decisions.
- Prepare, clean, and explore real-world datasets.
- Understand customer, campaign, and business metrics.
- Analyze relationships among marketing variables.
- Build preliminary predictive models.
- Evaluate model output and communicate uncertainty.
- Create analytically supported visualizations.
- Use AI assistants responsibly in analytics work.

- Translate findings into managerial recommendations.

The emphasis is practical, but not superficial. Students will learn technical concepts because those concepts matter for interpretation. They will use tools because tools make analysis possible. But the final goal is not a completed notebook, a model score, or a dashboard. The final goal is a recommendation that helps a decision-maker act with better evidence.

How This Guide Is Organized

This guide is organized into three parts. Chapters are assigned one per week and align with the session sequence in the course syllabus, so the reading for each session is the working material for that session.

Part I: Foundations of Marketing Analytics

Part I introduces analytics as a form of decision support. Students learn how to frame marketing problems, define variables, prepare data, explore datasets, and summarize customer and business patterns. These chapters establish the foundation for later modeling and visualization work.

Part II: Predictive Marketing Analytics

Part II focuses on methods used to understand relationships, segment customers, forecast outcomes, evaluate experiments, and build predictive models. The emphasis is not only on applying methods, but also on understanding what those methods mean, when they are useful, and how they can be misinterpreted.

Part III: Data Visualization and Analytics Communication

Part III focuses on using visualization to analyze data and communicate findings. Students learn how to select appropriate visual forms, avoid misleading designs, create dashboards, and turn analytical findings into executive-facing recommendations.

The guide also includes briefs and rubrics for the two major group projects in the course: the **Analytics Project (Project #1)**, an end-to-end predictive analytics or causal-evidence project, and the **Visualization Project (Project #2)**, an executive-facing Tableau dashboard or visual report. These projects ask students to apply the full analytics workflow to realistic marketing problems.

How Each Chapter Works

This guide is designed to be used one chapter at a time. Each chapter functions as a weekly learning unit and follows a consistent structure:

Marketing Decision Context. Each chapter begins with a practical marketing problem, connecting the technical topic to a decision that a manager, analyst, or marketing team might face.

Key Concepts. Each chapter introduces the core ideas needed to understand the method or workflow. Definitions are explained in plain language and tied to marketing examples.

Analytics Workflow. Each chapter walks through a practical analytic process. Depending on the topic, this may involve preparing data, summarizing a dataset, estimating a model, evaluating output, or designing a visualization.

AI-Augmented Analytics Lab. Each chapter includes an AI-assisted activity. Students use an AI tool to support part of the workflow, such as generating starter code, proposing chart types, explaining output, debugging an error, or drafting a managerial summary.

Verification Check. Each chapter includes checkpoints that ask students to verify the work. These checks may involve row counts, summary statistics, missing values, model diagnostics, chart scales, benchmark comparisons, or interpretation limits.

Managerial Interpretation. Each chapter asks students to translate analytic output into business language. The focus is on what the result means, why it matters, and what action it may support.

Reflection. Each chapter ends with reflection questions that help students connect the technique to professional practice, ethical judgment, and decision-making.

Companion Site and Course Files

The files and code used in this guide are available through the companion GitHub repository: https://github.com/jrmst102/businessanalytics

The companion site provides working materials for the weekly chapters, including Python notebooks, example datasets, code templates, verification exercises, project starter files, Tableau support files, and AI-use documentation templates when applicable.

Students should use the companion site alongside the assigned chapter. The guide explains the concepts and workflow; the repository provides the files needed to practice and complete the exercises. The repository is not a substitute for the chapter readings, and students should not submit repository files without adapting, explaining, and verifying them. The purpose of the companion site is to support analytic practice, not to provide finished answers.

Recommended Companion Site Workflow

1. Read the assigned chapter.
2. Open the corresponding notebook, dataset, Tableau file, or template.
3. Review the business question and data.
4. Specify the analytic task before prompting an AI assistant.
5. Run or adapt the code or visualization workflow.
6. Verify the output.
7. Interpret the result in managerial language.
8. Complete the assignment and AI-use documentation.

The AI-Augmented Analytics Workflow

This guide uses a four-part workflow: specify, predict-then-verify, explain, and document.

1. Specify

Before using an AI assistant, define the analytic task. A good specification identifies the business question, the unit of analysis, the relevant variables, the intended method, and the criteria for success. AI tools respond to the quality of the instructions they receive: a vague prompt often produces plausible-looking but unfocused work, while a precise specification improves the chance that the tool will produce something useful. Ask:

- What decision is being supported?
- What question must the analysis answer?
- What is the unit of analysis?
- Which variables are needed?
- What method is appropriate?
- What would a useful result look like?

2. Predict, Then Verify

Before running AI-generated code or accepting AI-generated output, predict what should happen. After running it, verify that the result makes sense. Verification is one of the most important habits in analytics: it protects against hidden mistakes, incorrect assumptions, and polished but inaccurate output. Common checks include:

- Row and column counts.
- Variable names and data types.
- Missing values and duplicate records.
- Summary statistics and known totals.
- Chart scales and labels.
- Model fit and performance metrics.
- Holdout results or baseline comparisons.
- Whether the conclusion follows from the evidence.

3. Explain

Students must be able to explain what each step does and why it matters. This includes code, transformations, visualizations, models, and recommendations. Being able to run an analysis is not the same as understanding it. Ask:

- What does this step do?
- Why is this method appropriate?
- What does the output mean?
- What does the output not mean?
- What should a marketing manager take away?

4. Document

Every major assignment and project requires documentation of AI use. Documentation records which tools were used, what prompts were submitted, what the tool produced, what was accepted or rejected, and how the final work was verified. Documentation is not an administrative formality; it is part of analytic accountability. The AI-Use Appendix asks:

- Which tool did I use?
- What did I ask it to do?
- What did it produce?
- What did I change?
- What did it get wrong?
- How did I verify the final result?
- What judgment did I apply?

Responsible Use of AI

AI assistants are expected tools in this guide, but they are not substitutes for learning.

Students may use AI tools to:

- Generate starter code.
- Debug errors.
- Explain Python, Tableau, or analytics concepts.
- Suggest analytic approaches.
- Draft preliminary chart ideas.
- Summarize source material.
- Improve the clarity of written explanations.
- Create first drafts of visualizations or dashboards.
- Translate technical findings into managerial language.

Students may not use AI tools to avoid understanding the work. The following practices are unacceptable:

- Submitting AI-generated work that cannot be explained.
- Submitting AI-generated code without checking whether it works correctly.
- Presenting AI-generated interpretation as one's own reasoning without review.
- Fabricating verification steps.
- Uploading confidential, proprietary, or personally identifiable data to external AI tools.
- Using AI tools during assessments designated as AI-free.
- Omitting required AI-use documentation.

AI may assist the analysis, but the student remains the analyst of record.

Required Tools

Google Colab. Google Colab is used for Python-based analytics. Students will use it to load data, clean datasets, summarize variables, create exploratory visualizations, build models, and evaluate results.

Tableau. Tableau is used for visual analytics, dashboard design, and executive-facing communication. Students will use Tableau Desktop and Tableau Prep in the visualization portion of the course and in the final visualization project.

AI assistants. Students are expected to use at least one general-purpose AI assistant and one code-capable assistant. Examples include ChatGPT, Claude, Gemini, GitHub Copilot, Colab AI features, and similar tools approved by the instructor.

Companion repository. The companion repository at https://github.com/jrmst102/businessanalytics provides files, notebooks, datasets, templates, and project materials used with this guide.

Course materials. Datasets, Colab notebooks, Tableau files, project templates, rubrics, and AI-use documentation templates are distributed through the course learning management system and, where appropriate, through the companion repository.

What Students Are Expected to Learn

By the end of this guide, students should be able to:

1. Prepare data for forecasting sales and marketing outcomes.
2. Analyze information to discover and explore meaningful relationships in datasets.
3. Develop strategic recommendations based on the interpretation of real-world datasets.
4. Produce analytically supported data visualizations to communicate sales and marketing opportunities.
5. Use AI assistants responsibly to support, verify, and communicate analytics work.

The fifth outcome does not replace the first four. It describes the environment in which the first four now operate.

How to Read This Guide

This guide should be read actively.

Do not treat the chapters as background readings to skim before class. Each chapter is a working session. Read with the dataset, notebook, or tool open. When a concept is introduced, test it. When code is shown, run it. When a prompt is suggested, revise it. When a chart is produced, critique it. When a recommendation is written, ask whether it follows from the evidence.

A useful reading routine is:

1. Identify the business problem.

2. Review the key concepts.
3. Work through the example or lab.
4. Use an AI assistant to extend, explain, or troubleshoot the work.
5. Verify the result manually.
6. Translate the finding into managerial language.
7. Complete the reflection or documentation activity.

The goal is not to memorize every command. The goal is to develop analytic judgment.

A Note on Code

This guide uses Python, but it does not assume that students already know how to code.

The purpose of code in this guide is to support analysis. Students should understand what the code does, how to run it, how to recognize when it fails, and how to interpret the output. AI assistants can help write or debug code, but students remain responsible for the result.

When working with code, focus on four questions:

1. What is the input?
2. What transformation, calculation, or model is being applied?
3. What is the output?
4. How do I know the output is correct?

This approach is more important than memorizing syntax.

A Note on Visualization

Data visualization is not decoration. It is a form of reasoning.

A good visualization helps the viewer see a pattern, compare values, understand a relationship, identify an exception, or make a decision. A poor visualization can obscure the truth, exaggerate a finding, or create false confidence.

This guide treats visualization in two ways. First, visualization is used for analysis: exploring data, finding patterns, and checking assumptions. Second, visualization is used for communication: presenting findings to decision-makers through charts, dashboards, and visual reports.

AI tools can now generate charts quickly. That makes visual judgment more important, not less. Students must learn to evaluate chart type, scale, labeling, hierarchy, color, annotation, and the connection between the visual and the business question.

A Note on Managerial Recommendations

The final step in business analytics is not the model, the chart, or the dashboard. The final step is the recommendation.

A recommendation should be clear, evidence-based, actionable, and appropriately qualified. It should tell the decision-maker what the analysis suggests, why it matters, what action should be considered, and what uncertainty remains.

A strong recommendation answers:

1. What did we find?
2. Why does it matter?
3. What should the organization do?
4. What are the risks or limitations?

5. What should be measured next?

Analytics becomes valuable when it improves decisions.

Acknowledgments

This guide grew out of teaching Business Analytics and Data Visualization to graduate students in the MS in Integrated Marketing program at the New York University School of Professional Studies.

I am grateful to the students whose questions, projects, challenges, and professional experiences shaped the design of this material. Their work made clear that analytics education must evolve with the tools now used in practice while preserving the core disciplines of reasoning, verification, and communication.

I also acknowledge the broader community of educators, analysts, researchers, and practitioners who continue to rethink how analytics should be taught in an AI-assisted world.

Statement on the Use of Generative AI and AI-Assisted Tools

Generative artificial intelligence and other AI-assisted tools were used during the research, writing, revision, and production of this textbook. Their uses included literature discovery, source organization, bibliographic management, outlining, development of preliminary drafts, revision of prose, review of consistency across chapters, support for code and analytical examples, and assistance with document formatting and visual presentation.

These tools were used under the author's direction and were not treated as authors, independent researchers, or authoritative sources. The author determined the scope, learning objectives, conceptual framework, chapter organization, analytical methods, interpretations, examples, and final recommendations presented in the book.

All AI-assisted material was reviewed, revised, and approved by the author. Factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries. Code, calculations, datasets, and analytical outputs were tested or otherwise reviewed for accuracy. Formatting and design suggestions were evaluated for readability, accessibility, and consistency with the book's design standards.

No AI system is credited as an author or contributor. Responsibility for the accuracy, originality, integrity, and final form of this textbook rests entirely with the author. No confidential student records, personally identifiable customer information, or other restricted institutional data were intentionally submitted to external generative AI systems during the preparation of this work.

Contents

Part I: Foundations of Marketing Analytics

Chapter 1. Becoming an AI-Augmented Marketing Analyst

Chapter 2. Marketing Analytics as Decision Support

Chapter 3. Data, Measurement, and Marketing Variables

Chapter 4. Data Preparation and Exploratory Analysis

Chapter 5. Descriptive Analytics and Customer Insight

Part II: Predictive Marketing Analytics

Chapter 6. Segmentation and Targeting Analytics

Chapter 7. Relationships, Drivers, and Regression

Chapter 8. Predictive Modeling for Marketing Decisions

Chapter 9. Classification, Propensity, and Churn Models

Chapter 10. Forecasting Demand, Sales, and Campaign Performance

Chapter 11. Experiments, A/B Testing, and Causal Evidence

Project #1. Analytics Project (Predictive Analytics or Causal Evidence)

Part III: Data Visualization and Analytics Communication

Chapter 12. Data Visualization for Analysis

Chapter 13. Data Visualization for Communication and Decision-Making

Project #2. Visualization Project

Appendices

Appendix A. Python and Google Colab Quick Reference

Appendix B. Common Marketing Metrics

Appendix C. AI Prompting Templates for Analytics

Appendix D. AI-Use Documentation Template

Appendix E. Data Visualization Checklist

Appendix F. Project Rubrics

Back Matter

Glossary

References

Chapter 01

Becoming an AI-Augmented Marketing Analyst

From Business Questions to Evidence-Based Marketing Decisions

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter presents marketing analytics as disciplined decision support and defines the role of the AI-augmented marketing analyst. It distinguishes business analytics, marketing analytics, and data-informed decision-making; explains descriptive, diagnostic, predictive, and prescriptive analytics; and shows how business questions become analytical questions. A seven-step workflow adapted from CRISP-DM is paired with four habits for accountable AI use: specify, predict-then-verify, explain, and document. The chapter also distinguishes artificial intelligence from generative AI assistants and reviews risks associated with generated text and code. A Google Colab lab introduces campaign data, basic marketing metrics, and verification checks. The final sections address managerial interpretation, industry practice, responsible analytics, and applied exercises.

KEYWORDS

marketing analytics; business analytics; artificial intelligence; generative artificial intelligence; data-informed decision-making; descriptive analytics; diagnostic analytics; predictive analytics; prescriptive analytics; verification

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Becoming an AI-augmented marketing analyst. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 1, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

1. Describe the role of the marketing analyst in data-rich and AI-assisted business environments.
2. Distinguish among descriptive, diagnostic, predictive, and prescriptive analytics in marketing contexts.
3. Explain how business questions, analytical questions, data, evidence, and recommendations are connected.
4. Distinguish artificial intelligence broadly from generative AI assistants, and identify where each may support marketing analytics and where human judgment remains necessary.
5. Use a basic Google Colab workflow to inspect a small marketing dataset.
6. Apply simple verification checks before accepting analytic output.
7. Translate analytic output into a cautious, evidence-based marketing recommendation.

CONCEPT

What This Chapter Is Really About

This chapter is not mainly about learning code. It is about learning the professional discipline of an analyst. Code, charts, models, dashboards, and AI tools are useful only when they help a decision-maker understand a problem and act with better evidence.

The guiding question is therefore: How does a marketing professional become a better decision-maker in a world where data and AI tools are widely available?

1.1 Marketing Decision Context: The Holiday Budget Decision

Imagine that you have just joined the marketing analytics team at StyleCraft Collective, a fictional mid-sized apparel retailer that sells through its website, mobile app, email program, and selected pop-up events. The company has a loyal customer base, but sales growth has slowed. The chief marketing officer wants the team to improve campaign performance before the holiday season, and she has asked for a recommendation in three weeks, before media commitments are finalized. Whatever the team recommends will determine where the holiday budget goes and which customers hear from the brand first.

At the weekly marketing performance review, each team brings a different view of the business. The email manager reports that open rates improved after a subject-line test. The paid media specialist argues that search ads generated the highest number of conversions. The social media manager points to engagement on short-form videos. The CRM manager worries that repeat purchase rates have declined among customers who bought only once. The finance team asks whether the marketing budget should be shifted away from low-margin promotions.

Everyone has data. Not everyone has the same question. Some participants want to describe what happened. Others want to understand why performance changed. Others want to predict what is likely to happen next. Still others want to decide what action the company should take. The review stalls because reports, opinions, metrics, and recommendations are being mixed before the decision has been clarified.

Notice what is missing. No one has stated the decision. Without a stated decision, every metric looks relevant and none is decisive. The improvement in open rates is real, but it does not tell the chief marketing officer whether to move money out of paid search. The finance team's margin concern is legitimate, but it does not identify which promotions destroy margin. Analytics cannot resolve the disagreement until someone names the choice that has to be made and the date by which it must be made.

There is also a constraint that will shape everything the team can do. Three weeks is enough time to describe what happened, compare a few segments, and assemble a defensible recommendation. It is not enough time to run a clean experiment, rebuild the customer data model, or wait for a quarter of new results. Part of the analyst's job is to say plainly what can and cannot be answered inside the time available, and then to propose the strongest evidence that fits. A recommendation that arrives after the media commitments are signed has no value, however rigorous it may be.

This situation is common. Marketing teams often have dashboards, campaign reports, CRM exports, web analytics, social media metrics, and customer feedback. However, the availability of data does not automatically produce better decisions. The value of analytics depends on how well the team frames the problem, prepares the data, selects the method, verifies the output, and communicates the recommendation.

Over the next three weeks, the team will likely use AI tools to compare channels and draft code for promotion analysis. Those uses are reasonable. The issue is whether the results are checked and whether the person signing the recommendation can explain how each number was produced. Accountability for the evidence remains with the analyst throughout the guide.

StyleCraft returns throughout the book as a running case. The company does not need analytics because it lacks data. It needs analytics because it must decide what to do.

> **CONCEPT**
>
> **Decision Context**
>
> In this guide, the decision context refers to the managerial situation in which an analysis will be used. It includes the decision to be made, the stakeholders, the available evidence, the constraints, the timing of the decision, and the consequences of action. This is an operational course concept rather than a standalone academic construct.

1.1.1 Opening Case Questions

1. What decision does StyleCraft need to make before the holiday season?
2. Which parts of the performance review are descriptive, diagnostic, predictive, or prescriptive?
3. What additional information would help the team decide whether to change the marketing budget?
4. Where might an AI tool help, and where would human judgment still be required?

1.2 What Does a Marketing Analyst Do?

A marketing analyst helps organizations make better decisions about customers, markets, campaigns, products, pricing, channels, and communication. This work may involve data preparation, descriptive summaries, visualization, predictive modeling, experiments, dashboards, and written recommendations. However, the analyst is not simply a person who produces charts or runs code. The analyst is responsible for disciplined reasoning.

Marketing analytics is especially challenging because marketing decisions are often made under uncertainty. Customers change their preferences. Competitors respond. Channels shift. Advertising platforms change their algorithms. Macroeconomic conditions affect spending. Metrics may be incomplete, biased, or difficult to compare. Even when a dataset is accurate, it may not fully represent the decision problem. As a result, marketing analysts must combine quantitative evidence with business context.

Furthermore, the modern analyst often works with AI tools. These tools can draft code, explain an error message, summarize customer comments, suggest visualizations, or create a preliminary managerial summary. These uses can save time. They do not remove the need for judgment. As Davenport, Guha, Grewal, and Bressgott (2020) argue, AI is likely to be most valuable in marketing when it augments, rather than replaces, human managers.

An AI-augmented analyst does not outsource thinking to software. The analyst defines the question, selects the evidence, checks the work, interprets the result, and takes responsibility for the recommendation.

1.2.1 The Analyst as Translator

One useful way to understand the marketing analyst is as a translator among three languages. The first is the language of business: revenue, customers, margins, budgets, campaigns, channels, and growth. The second is the language of data: rows, variables, distributions, missing values, relationships, models, and uncertainty. The third is the language of decision-making: alternatives, trade-offs, risks, recommendations, and next steps.

Many analyses fail because these languages are not connected. A technically correct model may not answer the manager's question. A polished dashboard may not clarify what action should be taken. A written recommendation may go beyond what the evidence supports. The analyst's task is to keep these languages aligned.

CONCEPT

Analytic Translation

Analytic translation is the process of converting a business problem into an analytical task, interpreting the analytical result in business language, and turning the evidence into a recommendation that a decision-maker can use.

In other words, the analyst does not merely produce output; the analyst connects output to action.

1.3 Business Analytics, Marketing Analytics, and Data-Informed Decisions

The following definitions connect the analyst's role to marketing decision support.

DEFINITION

Business Analytics

Business analytics is the use of data, quantitative methods, models, visualization, and technology to understand business problems and support decisions. In this guide, the phrase emphasizes decision support rather than technique for its own sake.

Source: Adapted from Camm et al. (2023), Davenport (2006), and Provost and Fawcett (2013).

DEFINITION

Marketing Analytics

Marketing analytics is the application of analytic methods, data, models, and technologies to marketing decisions. These decisions may involve customer understanding, campaign evaluation, segmentation, targeting, personalization, pricing, retention, forecasting, and marketing communication.

Source: Adapted from Wedel and Kannan (2016) and France and Ghose (2019).

The distinction matters because this guide is not a general statistics or programming text. It approaches analytics from the perspective of marketing decision-making. The central question is not simply whether an analysis can be performed, but whether the analysis helps a marketer understand a business problem and choose a more defensible course of action.

CONCEPT

Data, Information, Insight, and Recommendation

Data are recorded observations, values, or symbols. Information is data organized so that it becomes meaningful for a purpose. An insight is an interpretation that explains why a pattern matters. A recommendation is a proposed action supported by evidence, judgment, and context.

For example, a file containing 50,000 email recipients and clicks is data. A table showing that the click-through rate was 4.2 percent is information. The finding that recent purchasers clicked at a higher rate is an insight. The proposal to prioritize recent purchasers in the next email sequence is a recommendation.

Source: Adapted from Ackoff (1989), Rowley (2007), Farris et al. (2015), and Provost and Fawcett (2013).

1.3.1 Why Data-Informed, Not Data-Driven?

This guide uses data-informed because metrics and models do not determine actions on their own. Marketing decisions also require judgment about brand positioning, customer trust, ethics, competitive response, resource limits, and uncertainty.

For example, if a model predicts that a steep discount will increase conversions, a manager still needs to consider margin, customer expectations, brand equity, and whether frequent discounting trains customers to wait for promotions. Analytics can improve a decision without transferring responsibility for it.

CONCEPT

Data-Informed Decision-Making

Data-informed decision-making uses data and analysis as essential inputs into managerial judgment while recognizing that context, ethics, uncertainty, and strategic priorities also matter. This wording is used as an operational concept for this guide.

Source: Adapted from Provost and Fawcett (2013) and Davenport (2006).

1.4 The Four Types of Analytics

Introductory business analytics is often organized into descriptive, predictive, and prescriptive categories. This chapter adds diagnostic analytics because marketers often need to investigate why performance changed before recommending an action.

The four categories are related, but they are not interchangeable. A dashboard that reports last month's sales is not a forecast. A forecast is not a recommendation. A recommendation should use evidence, but it must also consider the decision context and the available alternatives. Table 1.1 summarizes the four types and their managerial value.

Table 1.1
Four Types of Analytics in Marketing

Type of analytics	Main question	Common outputs	Marketing example	Managerial value
Descriptive analytics	What happened?	Reports, dashboards, summary statistics, charts	Campaign conversion rate by channel last month	Clarifies performance
Diagnostic analytics	Why might it have happened?	Comparisons, drill-downs, cohort analysis, segment analysis	Lower conversion among first-time visitors than returning customers	Identifies possible drivers
Predictive analytics	What is likely to happen?	Forecasts, scores, probabilities, predictive models	Probability that a customer will churn in the next 30 days	Anticipates outcomes

Type of analytics	Main question	Common outputs	Marketing example	Managerial value
Prescriptive analytics	What should we do?	Decision rules, optimization, recommendations, action plans	Allocate budget to campaigns with the highest expected incremental contribution	Supports action

DEFINITION

Descriptive Analytics

Descriptive analytics summarizes past or current data to show what happened. In marketing, it may include sales reports, campaign dashboards, customer profiles, channel comparisons, and summary statistics.

Source: Adapted from Camm et al. (2023) and Farris et al. (2015).

CONCEPT

Diagnostic Analytics

Diagnostic analytics investigates why performance may have changed by comparing groups, segments, time periods, or possible drivers. In marketing, it may involve cohort analysis, segment comparison, or channel drill-downs.

Caution: diagnostic analytics can identify plausible drivers and narrow the investigation, but observational comparisons do not by themselves establish causality.

Source: Course concept developed for this guide. The comparative methods it relies on are discussed in Camm et al. (2023) and France and Ghose (2019).

That caution deserves emphasis, because diagnostic language invites causal claims. A drill-down showing lower conversion among first-time visitors has identified an association, not a cause. Those visitors may differ in intent, in traffic source, in device, or in what they were shopping for. Chapter 7 examines how regression describes relationships without demonstrating cause, and Chapter 11 shows what kind of evidence a causal claim actually requires. Until then, treat every diagnostic finding as a candidate explanation to be tested rather than a conclusion to be reported.

DEFINITION

Predictive Analytics

Predictive analytics uses patterns in existing data to estimate what is likely to happen for new or future observations. In marketing, it may include churn prediction, sales forecasting, lead scoring, response modeling, or conversion probability estimation.

Predicting an outcome well and explaining what causes it are different goals, and a model that does one may do the other poorly.

Source: Adapted from Shmueli (2010) and Provost and Fawcett (2013).

DEFINITION

Prescriptive Analytics

Prescriptive analytics uses evidence, models, constraints, and judgment to recommend actions. In marketing, it may support budget allocation, targeting, offer selection, pricing, promotion planning, or resource prioritization.

Source: Adapted from Camm et al. (2023) and Davenport (2006).

1.4.1 Example: Email Campaign Performance

Suppose StyleCraft sent three campaigns: a new-arrivals campaign, a loyalty discount campaign, and a clearance campaign. The descriptive question is: Which campaign had the highest open rate, click-through rate, conversion rate, and revenue? The diagnostic question is perhaps the most interesting: Why did the loyalty campaign perform better among returning customers but worse among new customers? The predictive question is: Which customers are most likely to respond to the next email? The prescriptive question is: Which customers should receive which offer next week, given budget limits and margin goals?

The same dataset may support all four types of analysis, but each type asks a different question. One of the analyst's first responsibilities is to identify which question is being asked.

1.5 From Business Question to Analytical Question

Analytics begins with the business question, not the dataset. A business question is the managerial issue that the organization needs to address; an analytical question is the measurable version of that issue. The business question states what must be decided; the analytical question states what must be measured.

Consider the question, "How can we improve customer loyalty?" This is a reasonable business concern, but it is too broad for immediate analysis. An analyst might translate it into more specific analytical questions: Which customer segments have the highest repeat purchase rates? Which first-purchase categories are associated with a second purchase? How long does it typically take a new customer to make a second purchase? Which campaign contacts are associated with retention? Each analytical question points to different data and methods. Table 1.2 illustrates this translation for several common marketing questions.

Table 1.2
Translating Business Questions into Analytical Questions

Business question	Possible analytical question	Potential data needed	Possible decision supported
Should we increase paid search spending?	Which paid search campaigns generate the highest conversion rate and contribution margin?	Campaign costs, clicks, conversions, order value, margin	Budget allocation
Why are repeat purchases declining?	Which customer cohorts show the largest decline in second-purchase rate?	Customer IDs, purchase dates, order sequence, product categories	Retention campaign design
Which customers should receive a discount?	Which customers have high predicted response but low likelihood of buying without an offer?	Purchase history, engagement, prior discounts, margins	Offer targeting
Is the loyalty program working?	Did members show higher repeat purchase or average order value after enrollment than comparable nonmembers?	Enrollment date, purchases, customer attributes, time periods	Program continuation or redesign

Business question	Possible analytical question	Potential data needed	Possible decision supported
What should the dashboard emphasize?	Which metrics best indicate whether campaign performance is improving against business goals?	KPIs, targets, historical performance, stakeholder needs	Executive reporting design

CONCEPT

Business Questions and Analytical Questions

A business question identifies the decision or managerial issue. An analytical question identifies what must be measured, compared, modeled, or visualized to support that decision. Strong analytics requires both.

Source: Adapted from Provost and Fawcett (2013) and Chapman et al. (2000).

1.6 The Marketing Analytics Workflow

CRISP-DM, a widely cited data mining process, begins with business understanding, then moves through data understanding, data preparation, modeling, evaluation, and deployment (Chapman et al., 2000). This guide adapts that logic for marketing students and AI-assisted practice.

The adapted workflow has seven steps: define the marketing decision, translate the decision into analytical questions, identify and prepare the data, explore patterns, apply an appropriate method, verify and interpret the output, and communicate a recommendation. Table 1.3 presents the seven steps and the analyst's work at each one. Table 1.4 then shows, for the same seven steps, where AI assistance typically helps and what must be verified before the step is trusted.

Table 1.3
A Marketing Analytics Workflow: Steps and Analyst Work

Step	Question to ask	Typical analyst activity
1. Define the decision	What decision needs support?	Clarify decision, audience, timing, constraints
2. Translate the question	What can be measured or modeled?	Define unit of analysis, variables, outcome
3. Prepare the data	Is the data usable?	Clean, join, filter, reshape, document
4. Explore patterns	What does the data suggest?	Summaries, charts, segments, comparisons
5. Apply a method	What method fits the question?	Regression, classification, forecasting, clustering, A/B test, dashboard
6. Interpret evidence	What does the output mean?	Translate output into business language
7. Recommend action	What should be done next?	Write a clear, qualified recommendation

Table 1.4
A Marketing Analytics Workflow: AI Support and Verification Checks

Step	Where AI may help	Verification check
1. Define the decision	Draft a problem statement	Confirm the decision and deadline with the stakeholder
2. Translate the question	Suggest candidate analytical questions	Check that each question maps to a decision
3. Prepare the data	Generate starter cleaning code	Check rows, columns, missing values, duplicates
4. Explore patterns	Suggest charts or summary code	Check scales, labels, totals

Step	Where AI may help	Verification check
5. Apply a method	Draft model or chart code	Compare the result with a baseline or a stated expectation
6. Interpret evidence	Draft a plain-language explanation	Check whether the claims follow from the evidence
7. Recommend action	Draft a summary or memo	Identify risks, limits, and the next measurement

CONCEPT

Verification

Verification is the practice of checking whether analytic output is accurate, reasonable, and aligned with the analytical question before using it in a recommendation. Verification is especially important when code, charts, or summaries are produced with AI support. In this guide, verification is treated as a professional habit rather than as a single statistical test.

Source: Adapted from Chapman et al. (2000) and Provost and Fawcett (2013).

1.7 The AI-Assisted Workflow Used in This Guide

Within the seven-step workflow, AI use follows four habits: specify, predict-then-verify, explain, and document. Together, they make AI-assisted work accountable.

First, specify the task. Before asking for help, define the business question, what one row represents, the relevant variables, the intended output, the constraints, and the verification criteria. A vague prompt such as "analyze this data" is unlikely to produce a useful result. A stronger prompt tells the tool what the data represent and what decision the analysis should support.

Second, predict-then-verify. Before running code or accepting an interpretation, write down what you expect the result to be: a number, range, shape, or direction. Then compare the actual output with that expectation. If they do not match, stop and investigate. Either the output is wrong, your expectation is wrong, or the data contain something that requires explanation.

Third, explain the work. The analyst should be able to explain what each major step does and what the output means. Fourth, document the use of AI. Documentation should record the tool used, the prompt, the output, the revisions made, the errors or limitations found, and the verification performed. The AI-use documentation template in Appendix D is the standard form for that record throughout this guide.

CONCEPT

Specify, Predict-Then-Verify, Explain, Document

Specify: define the task before asking for help.

Predict-then-verify: write your expectation, run the step, compare the output with the expectation, and investigate any mismatch.

Explain: be able to explain what each step does and what the result means.

Document: record how AI was used and how the work was checked.

Source: Author-developed framework for this guide, informed by the analytic process logic in Chapman et al. (2000) and Provost and Fawcett (2013) and by the AI-augmentation argument in Davenport et al. (2020).

1.7.1 Example: A Weak Prompt and a Stronger Prompt

The difference between the two prompts below is the specify step. Neither prompt requires special technique; the second simply says what the data are and what the analysis is for.

A weak prompt. “Analyze this campaign data and tell me what to do.”

A stronger prompt. “I am analyzing a marketing campaign dataset with one row per campaign. The columns are campaign_name, channel, impressions, clicks, conversions, revenue, and ad_spend, where ad_spend is advertising spend only. Please write pandas code to calculate click-through rate, post-click conversion rate (conversions divided by clicks), cost per conversion, and return on ad spend by campaign. Do not interpret the results yet. Include checks for missing values and for zero denominators.”

The second prompt is better because it specifies what one row represents, the variables, the calculations, the denominator for each rate, the software context, and the limits on what the assistant should do. It also asks for verification before interpretation.

1.8 Key Marketing Data Concepts

Before the first lab, students need a small set of data concepts. Chapter 3 develops them formally; here they are introduced only as working ideas.

Begin with one question: what does one row represent? In the lab that follows, each row is one marketing campaign. In a customer file, each row is one customer. In a transaction file, each row is one purchase. The answer matters because it determines what a total, an average, or a rate can legitimately describe. Unless those quantities have been aggregated into explicit columns, a campaign-level table does not reveal individual customers, and a customer-level table does not reveal individual transactions. A surprising share of analytic errors are simply one kind of row being interpreted as if it were another.

Marketing data can come from CRM systems, point-of-sale files, websites, email platforms, social media platforms, advertising accounts, surveys, loyalty programs, customer service interactions, experiments, and third-party data providers. Each source has strengths and limitations.

Because marketing data are imperfect, analysts should not treat datasets as neutral mirrors of reality. A dataset is a representation of business activity shaped by what the organization measured, what systems captured, what customers consented to share, and what processing occurred before the analyst received the file. Whether a particular dataset is fit for a particular purpose is a question this guide returns to repeatedly.

Chapter 3 develops these concepts formally. It defines the unit of analysis and the grain of a table, separates stored data types from levels of measurement, specifies the four elements of a ratio metric, and sets out the dimensions along which data quality is assessed. For now, one habit is enough: before analyzing any file, state what one row represents, and write it down where the reader of your analysis can see it.

1.9 AI in Marketing Analytics: Benefits and Risks

Davenport et al. (2020) and Huang and Rust (2021) describe broad applications of AI across marketing research, strategy, and action. This course relies mainly on generative AI assistants, which have a different risk profile from predictive and algorithmic systems.

> **CONCEPT**
>
> **Artificial Intelligence and Generative Artificial Intelligence**
>
> Artificial intelligence is the broad category: computational systems that perform tasks associated with human intelligence, including pattern recognition, prediction, classification, and language processing.
>
> Generative artificial intelligence refers to systems that produce new content such as text, code, images, audio, or video, in response to a prompt.
>
> In this course, most AI-assisted exercises use generative-AI assistants such as chatbots and notebook copilots. Later chapters also examine predictive and algorithmic systems, including the scoring and targeting models that marketing organizations run without generating any text at all.

Source: Adapted from Davenport et al. (2020), Huang and Rust (2021), and National Institute of Standards and Technology (2024).

For students, the most immediate use of AI may be as a workflow aid. A tool can explain why Python code failed, suggest a pandas command, summarize the meaning of an output, or propose chart alternatives. These tools can speed up learning and routine work, but students still need to understand and verify the output.

The risks are practical, and most of the ones that will affect your coursework belong specifically to generative AI rather than to artificial intelligence in general. A generative assistant may invent sources, misread columns, assume the wrong unit of analysis, ignore missing values, or generate code that runs cleanly while computing the wrong metric. The federal risk framework for generative AI names this failure directly as confabulation, the generation and confident presentation of erroneous content in response to a prompt (National Institute of Standards and Technology, 2024). That framework also identifies data privacy, intellectual property, information integrity, and human–AI configuration as risk areas, all of which apply when confidential material is placed into an external system or when generated output is passed along unchecked. Responsible use therefore requires the analyst to supervise the assistant. Table 1.5 summarizes where AI support is useful across the workflow and what the analyst remains responsible for.

Table 1.5
AI Support and Risks Across the Marketing Analytics Workflow

Workflow step	Useful AI support	Risk to watch for	Analyst responsibility
Framing the problem	Suggest possible analytical questions	Questions may be generic or misaligned	Select questions based on the decision
Preparing data	Generate cleaning code or explain errors	Code may remove important records or mishandle missing values	Check row counts, logic, and assumptions
Exploring data	Suggest summaries or chart types	Charts may be decorative rather than useful	Match visuals to the question
Modeling	Draft starter code	Method may be inappropriate or poorly evaluated	Choose the method and compare baselines
Interpretation	Draft plain-language explanation	Interpretation may overclaim causality or certainty	Qualify conclusions
Recommendation	Draft a memo or summary	Recommendation may ignore constraints or ethics	Make the final judgment

CONCEPT

AI in Marketing Analytics

In this guide, AI in marketing analytics refers to the use of computational systems to automate tasks, process data, identify patterns, support predictions, personalize actions, or assist communication in marketing research, strategy, and execution.

Source: Adapted from Davenport et al. (2020) and Huang and Rust (2021).

AI IN PRACTICE

Review Before You Rely

A useful professional standard is simple: do not send work to a manager that you cannot explain. AI can help draft code, summarize output, or improve a first version of a memo. However, the analyst must still check the work, revise it, explain it, and stand behind the recommendation.

Audit routine. After an assistant returns code or an interpretation, ask four questions in order. Does the output answer the question I specified? Does one row still represent what I said it represents? Can I reproduce at least one number by hand? Can I state in one sentence what this result does not show?

If any answer is no, the work is not ready to leave your desk. When it is ready, record the tool, the prompt, the revisions, and the checks using the AI-use documentation template in Appendix D.

1.10 Hands-On Application in Python and Google Colab

This first hands-on activity is intentionally simple: open a notebook, run a code cell, inspect a small marketing dataset, and ask basic questions about it. Later chapters build on these skills.

Google Colab is a browser-based notebook environment. In this guide, notebooks combine explanation, code, output, tables, and interpretation in one place. This format is useful because the reader can connect reasoning with computation. Students who have not used a notebook before should work through the setup steps and the short Python reference in Appendix A before starting Lab 1.1.

1.10.1 Lab 1.1: Inspecting a Small Campaign Dataset

In this lab, you will create a small fictional campaign dataset directly in Python. Later chapters will use larger datasets from the companion repository. This first lab avoids file-loading issues so that the focus can remain on the workflow. The numbers are simplified for instruction and should not be treated as industry benchmarks. Note the column named ad_spend: it records advertising spend for the campaign, not total campaign cost. That distinction matters as soon as return on ad spend is calculated.

Code 1.1. Create and inspect a small marketing campaign dataset

```
import pandas as pd

campaigns = pd.DataFrame({
    "campaign": [
        "New Arrivals", "Loyalty Offer",
        "Clearance", "Welcome Series",
    ],
    "channel": [
        "Email", "Email", "Paid Search", "Social",
    ],
    "impressions": [52000, 41000, 68000, 75000],
    "clicks": [2860, 2540, 3400, 3000],
    "conversions": [310, 420, 390, 250],
    "revenue": [27900, 33600, 23400, 17500],
    "ad_spend": [1800, 1200, 9500, 5200],
})

campaigns.head()
```

The code begins by importing pandas, a Python library commonly used for data analysis. It then creates a small table called campaigns. Each row represents one marketing campaign. The columns describe the campaign name, channel, impressions, clicks, conversions, revenue, and advertising spend.

CONCEPT

Input, Transformation, Output

Input is the data or object used in a step. A transformation is the calculation, filter, model, or change applied to the input. Output is the result produced by the step.

> A good analyst can identify the input, the transformation, and the output for each major step of an analysis.

1.10.2 Adding Basic Marketing Metrics

The next code cell creates three common campaign metrics. Click-through rate divides clicks by impressions. Post-click conversion rate divides conversions by clicks. Return on ad spend divides revenue by advertising spend.

Each of those sentences hides a choice. Conversion rate in particular has no single denominator: depending on the question, conversions may be divided by clicks, sessions, visitors, leads, or recipients, and the four results will differ. For this lab, post-click conversion rate is defined as conversions divided by clicks, and the column is named accordingly so that the definition travels with the number. Return on ad spend divides revenue by advertising spend, which is why the spend column here is named ad_spend rather than cost. A total campaign cost that also included creative production or agency fees would produce a different ratio, and one that is not comparable with the industry convention for this metric.

Code 1.2. Calculate basic campaign metrics

```
campaigns["click_through_rate"] = (
    campaigns["clicks"] / campaigns["impressions"]
)
campaigns["post_click_conversion_rate"] = (
    campaigns["conversions"] / campaigns["clicks"]
)
campaigns["roas"] = (
    campaigns["revenue"] / campaigns["ad_spend"]
)

campaigns[[
    "campaign", "channel", "click_through_rate",
    "post_click_conversion_rate", "roas",
]]
```

These calculations are simple, but they illustrate an important principle: metrics are constructed. The analyst chooses a numerator and a denominator. If either is wrong, the metric will be wrong even if the code runs successfully. In other words, a metric is a choice made by the analyst, not a fact discovered in the data.

> **CONCEPT**
>
> **Metrics Are Constructed**
>
> Every metric is built, not found. The analyst chooses what goes in the numerator, what goes in the denominator, over what period, and for which records. Change any one of those choices and the number changes, even though the label on the column stays the same.
>
> Two campaign reports can therefore show different conversion rates without either being wrong. The habit this guide asks for is to write the definition down beside the number, every time.
>
> *Source: Course concept developed for this guide, informed by Farris et al. (2015) and Rust et al. (2004). Chapter 3 specifies the elements of a ratio metric formally.*

1.10.3 Verification Checks for Lab 1.1

Before interpretation, check the dataset's structure, missing values, denominators, rate ranges, and distributions. The first two checks print inspection results; the denominator test and the assertion stop the notebook when a required condition fails.

One note on order. The zero-denominator check logically belongs before the division in Code 1.2, and that is where it should sit in your own notebook. It appears here so that all of the chapter's verification logic can be read in one place.

Code 1.3. Run basic verification checks

```
# 1. Structure: how many rows and columns?
print("rows and columns:", campaigns.shape)

# 2. Missing values in any column
print(campaigns.isna().sum())

# 3. No required denominator may be zero
denominators = campaigns[
    ["impressions", "clicks", "ad_spend"]
]
if (denominators == 0).any().any():
    raise ValueError(
        "A required metric denominator contains zero."
    )

# 4. Rates expressed as proportions must fall in [0, 1]
rate_columns = [
    "click_through_rate",
    "post_click_conversion_rate",
]
rates = campaigns[rate_columns]
assert ((rates >= 0) & (rates <= 1)).all().all()

# 5. Inspect the distribution of the two rates
campaigns[rate_columns].describe()
```

VERIFICATION CHECK

Before you run: predict the number of rows and columns that campaigns.shape will report now that three metric columns have been added, and predict the range in which each rate must fall.

After you run: the frame should have 4 rows and 10 columns. Compare that with your prediction. A click-through rate or a post-click conversion rate expressed as a proportion must fall between 0 and 1, and the assertion in Code 1.3 will stop the notebook if one does not.

Investigate if: a rate is greater than 1, or the assertion fails, or the denominator check raises an error. The numerator and denominator may be reversed, the data may contain errors, or the metric may not be defined the way you assumed. Verification protects the analyst from interpreting faulty output, and an assertion protects the analyst from failing to notice.

1.11 Marketing Interpretation and Managerial Insight

Campaign metrics are not recommendations. Interpretation begins by asking what each metric suggests, which metric matters for the decision, and what trade-offs or limitations remain. Does a campaign with high revenue also have high spend? Does a high post-click conversion rate translate into high contribution?

Suppose the Loyalty Offer has the highest post-click conversion rate and the highest ROAS. A weak interpretation would say, "The Loyalty Offer is best." A stronger interpretation would say, "Among these four campaigns, the Loyalty Offer produced the strongest combination of conversion efficiency and return on ad spend.

However, the data do not show whether customers would have purchased without the offer, whether the discount reduced margin, or whether the campaign can scale to a larger audience."

The stronger interpretation is better because it connects the result to a decision while acknowledging uncertainty. It does not confuse performance measurement with causal proof. It does not assume that the best campaign in a small dataset should automatically receive the full budget. It identifies what the evidence supports and what remains unknown.

1.11.1 Revisiting the StyleCraft Case

The StyleCraft team should begin by clarifying the holiday-season decision. If the decision is budget allocation, the team needs campaign spend, revenue, margin, incremental lift, audience size, and operational constraints. If the decision is customer targeting, the team needs customer-level response and purchase history. If the decision is message strategy, the team may need creative performance, segment behavior, and customer feedback.

Descriptive analytics could summarize past campaign performance. Diagnostic analytics could compare performance across channels, segments, customer cohorts, or offer types. Predictive analytics could estimate which customers are most likely to respond to future campaigns. Prescriptive analytics could recommend how to allocate budget or which offers to send to which customers. AI tools could help with code, summaries, and drafting. The team must still verify the results, evaluate the assumptions, and make the final recommendation.

1.12 Business Analytics in Practice

This section examines how the chapter's habits appear in real organizations: how AI changes analysts' work, what happens when a deliverable is not verified, and what employers test in analytics interviews.

1.12.1 When the Coding Gets Easier, the Framing Gets Harder

Consider a consumer-packaged-goods insights team of eight analysts supporting brand and shopper marketing. Two years ago, a routine request, weekly promotion performance by retailer and pack size, took an analyst most of a day: pull the data, reshape it, build the summary, format the deck. Today an assistant drafts the query and the chart code in minutes. The time saved on coding shifted to other decisions: which retailers belong in the comparison, whether the promotion window should be defined by ship date or scan date, and whether the lift being reported is lift against last year, against the non-promoted weeks, or against nothing at all.

That redistribution of effort is consistent with what field experiments have found. In a preregistered study of 758 consultants at Boston Consulting Group, participants with access to GPT-4 completed 12.2 percent more tasks, completed them 25.1 percent more quickly, and produced work of measurably higher quality. On a task deliberately designed to fall outside the model's competence, however, participants using AI were 19 percentage points less likely to produce correct solutions than participants working without it (Dell'Acqua et al., 2026). The authors describe a jagged frontier: capability that is uneven in ways not visible from the outside. Because incorrect output can be fluent and polished, teams cannot judge reliability from presentation quality.

Teams that adapt well tend to do two things. They shift senior time toward problem framing, because framing decisions are the ones the assistant cannot make. And they make quality assurance an explicit, named step rather than something that happens if there is time. On the team described here, no recurring report leaves the group until a second analyst has confirmed the denominator, the date window, and the filter set. A review of that kind costs minutes and catches the errors a single analyst working quickly does not. As production becomes cheaper, specifying the question and checking the answer become the scarce skills.

1.12.2 The Deliverable That Could Not Be Verified

The failure mode is easiest to see in professional services, where the deliverable is the product. In October 2025, Deloitte Australia agreed to issue a partial refund on a report it had produced for the Australian government under a contract valued at approximately A$440,000, after the published version was found to contain references to academic papers that do not exist and a fabricated quotation attributed to a federal court judgment (Associated Press,

2025). The firm maintained that the substance of the review held. The citations, however, had been invented, and no one had checked them before publication.

Marketing analytics fails the same way, usually with less publicity. An agency deliverable arrives carrying a confident summary metric, say "incremental reach efficiency," that no one on the client side can reproduce, because the assistant that drafted the deck synthesized a plausible-sounding measure rather than computing a defined one. The client cannot verify it. The agency analyst cannot fully explain it. The metric nonetheless enters the next planning cycle, and by the time someone asks where the number came from, three decisions have been built on top of it.

The federal risk framework for generative AI gives this behavior a name: confabulation, the confident presentation of erroneous content (National Institute of Standards and Technology, 2024). The naming matters, because saying that the tool made a mistake locates the problem in the software. In the cases above, the software did what generative systems do. What was missing was a person who checked. An unverifiable number is a liability owned by the person who signs the deliverable.

1.12.3 What Hiring Managers Can Test For

Hiring has moved in the same direction. Employers identify analytical thinking as the most commonly required core skill, cited by seven in ten organizations, while AI and big data rank as the fastest-growing skill through 2030 (World Economic Forum, 2025). Employers therefore need analysts who can use AI tools and judge their output.

The verification habit is now common enough to be measured. In a survey of more than 33,000 respondents, 84 percent reported using or planning to use AI tools, yet only about 3 percent expressed high trust in the accuracy of what those tools produced. Sixty-six percent named "AI solutions that are almost right, but not quite" as their leading frustration, and roughly 45 percent reported that debugging AI-generated code took more time than expected (Stack Overflow, 2025). Analysts are not developers, but the pattern transfers directly: heavy use, low trust, and real time spent on the gap between the two.

That gap is what an interview can usefully probe. Instead of asking a candidate to write code from scratch, an interviewer might hand over a short notebook or a one-page analysis and ask what is wrong with it: a reversed denominator, a silently dropped set of rows, a rate above 1, a chart whose axis begins at 40 rather than 0. A candidate who can describe how they check work, and who can name an occasion when an assistant's output looked right and was not, is demonstrating exactly the judgment the survey evidence above suggests is scarce. Verification is therefore part of the analyst's work, not a final defensive step.

1.12.4 In Your First Analyst Job

The course mirrors the accountability structure you will work inside: labs require a prediction before execution, AI-assisted exercises require documentation of the tool, the prompt, the revision, and the check, and recommendation exercises require one limitation and one next step. Work in this course is assessed on whether it can be defended, not on whether the code ran. That is the same standard that will apply when the deliverable carries your name and the person reading it has a budget to spend.

1.13 Ethics, Privacy, and Responsible Analytics

Marketing analytics often uses customer data, creating legal, professional, and strategic responsibilities. Analysts should consider whether the data are appropriate to use, whether customers would reasonably expect the use, whether sensitive information is involved, whether the analysis could create unfair treatment, and whether the recommendation protects customer trust.

AI also creates specific privacy and governance risks. Uploading confidential customer data into an external AI system may violate company policy, privacy rules, or customer expectations, and data privacy is one of the risk areas that generative-AI governance frameworks specifically flag (National Institute of Standards and Technology, 2024). In this course, confidential or personally identifying data are never to be placed into an external AI tool. Model recommendations may also reproduce patterns of exclusion if historical data reflect unequal treatment. A responsible analyst asks not only, "Does the model work?" but also, "Should we use this result in this way?"

CONCEPT

Responsible Analytics

Responsible analytics is the practice of using data, methods, AI tools, and recommendations in ways that are accurate, transparent, privacy-aware, fair, and accountable to the decision context.

Source: Author-developed definition for this guide, synthesizing the marketing-privacy, data-ethics, and algorithmic-accountability literatures represented by Martin and Murphy (2017), Barocas and Selbst (2016), Mittelstadt et al. (2016), and Floridi and Taddeo (2016).

1.14 Chapter Summary

This chapter introduced the role of the AI-augmented marketing analyst. The main point is that analytics is a disciplined approach to decision support. It begins with a marketing decision, translates that decision into analytical questions, uses data and methods to generate evidence, verifies the output, and communicates a recommendation.

The chapter distinguished data, information, insight, and recommendation; introduced the four types of analytics and the caution that diagnostic comparisons do not establish cause; separated artificial intelligence broadly from the generative assistants students will use most; and presented the four habits of specify, predict-then-verify, explain, and document, which were then practiced in a short Google Colab lab. It also stepped outside StyleCraft to look at how those habits appear in an insights team, in a professional-services failure, and in analytics hiring.

The rest of this guide builds from this foundation. Later chapters examine marketing data, measurement, data preparation, exploratory analysis, segmentation, regression, predictive modeling, forecasting, experiments, visualization, dashboards, and storytelling. The tools will become more sophisticated, but the principle remains the same: analytics becomes valuable when it improves decisions. The next chapter examines marketing analytics as decision support in greater depth, showing how a decision is structured, how a vague request becomes a written specification, and how the deliverable is matched to the choice being made. Chapter 3 then turns to data, measurement, and marketing variables, where the concepts this chapter treated informally are defined in full.

1.15 Exercises for Practice and Homework

These exercises practice four habits: classify the analytic question, state what one row represents, verify the output, and connect the result to a marketing decision. Core chapter practice is required, in-class activities support discussion, and extensions are optional. Each exercise carries an assignment label so that instructors can assign selectively.

1.15.1 Core Chapter Practice

Exercise 1.1 Concept Check (Required Practice)

1. Explain the difference among data, information, insight, and recommendation.
2. What is the difference between a business question and an analytical question?
3. Why might data-informed be a better phrase than data-driven for many marketing decisions?
4. A dataset has one row for each customer purchase. What does one row represent, and what can that table not tell you?
5. Why is verification especially important when using AI-assisted code or interpretation?
6. What is the difference between artificial intelligence and generative artificial intelligence, and why does the distinction matter for the risks discussed in Section 1.9?

Exercise 1.2 Classify the Analytics Type (Required Practice)

For each scenario, identify whether the example is primarily descriptive, diagnostic, predictive, or prescriptive. Then explain your reasoning in one sentence.

1. A dashboard shows monthly revenue by marketing channel for the past twelve months.
2. A model estimates which customers have the highest probability of canceling a subscription next month.
3. An analysis compares conversion rates by device type and finds that mobile users abandon the checkout page more often.
4. A system recommends sending different retention offers to different customer segments based on expected profit.
5. A report shows that average order value increased after a new free-shipping threshold was introduced.
6. A forecast estimates expected weekly demand for the next quarter.
7. An analyst recommends reducing spend on a campaign because its cost per acquisition exceeds the target margin.
8. A chart shows that repeat purchase rates are lower among customers acquired through one promotion.

Exercise 1.3 Match the Metric to the Question (Required Practice)

For each business question, explain why the suggested metric is useful, state the denominator the metric requires, and identify one thing the metric does not tell you.

1. Are people engaging with our email? Suggested metric: click-through rate.
2. Are clicks turning into purchases? Suggested metric: post-click conversion rate.
3. Are we spending efficiently? Suggested metric: cost per conversion.
4. Are we generating revenue relative to advertising spend? Suggested metric: ROAS.
5. Are we retaining customers? Suggested metric: repeat purchase rate.
6. Are customers becoming more valuable over time? Suggested metric: customer lifetime value.

Exercise 1.4 Hands-On Colab Practice (Homework Submission)

1. Open a new Google Colab notebook.
2. Copy the code from Lab 1.1 into the notebook and run it.
3. Add a new column called cost_per_conversion that divides ad_spend by conversions.
4. Sort the campaigns from highest to lowest ROAS.
5. Write a three-sentence interpretation of the results.
6. Write one verification check that would make you more confident in the output, and run it.
7. Write one limitation of the dataset.

Exercise 1.5 Verification Checklist (Required Practice)

1. List one verification check for dataset structure.
2. List one verification check for missing or unusual values.
3. List one verification check for a zero or implausible denominator.
4. List one verification check for metric formulas.
5. List one verification check for interpretation limits.
6. List one verification check for AI-assisted output.
7. Which check do you think is most important for the StyleCraft case, and why?

Exercise 1.6 AI-Assisted Practice (Homework Submission)

1. Use an AI tool to support one part of Exercise 1.4, but do not allow the tool to replace your own interpretation.
2. Complete the compact documentation record shown in Table 1.6 in your notebook, then complete the full AI-use template in Appendix D for your submission.
3. State one thing the assistant produced that you revised or rejected, and explain how you detected it.

Table 1.6
AI-Use Documentation: Compact Record

Question	Student response
Which AI tool did you use, and for which step?	
What prompt did you submit?	
What did you accept, revise, or reject?	
How did you verify the final result?	

Table 1.6 is a compact record for use inside a notebook. The full AI-use documentation template, including fields for the output received, the errors or limitations found, and what you learned from the process, is in Appendix D and is the version to submit with graded work throughout this guide.

Exercise 1.7 Managerial Memo (Homework Submission)

Write a short memo of 200–300 words to the StyleCraft chief marketing officer. The memo should answer this question: What should the team examine before reallocating the holiday campaign budget? Include one descriptive question, one diagnostic question, one predictive question, one prescriptive question, and one caution about AI-assisted analysis. For the diagnostic question, state explicitly what it could and could not establish. Recommended memo structure: decision context, evidence needed, caution or limitation, and next step.

1.15.2 In-Class Activities

Exercise 1.8 Identify the Decision First (In-Class Discussion)

For each situation, answer: What decision is the manager trying to make? What evidence would help? What type of analytics would be most useful first?

1. Website traffic increased last month, but online sales did not.
2. A loyalty campaign generated many purchases, but the finance team is concerned about margin.
3. A paid social campaign produced many clicks, but few new customers.
4. Customer service complaints increased after a product launch.
5. A dashboard shows that one region is underperforming.

Exercise 1.9 Business Question Translation (In-Class Discussion)

Translate each business question into at least two analytical questions. State what one row would represent in the data you would need.

1. How can we improve the performance of our loyalty program?
2. Should we increase our investment in influencer marketing?
3. Why are customers abandoning their carts?
4. Which customers should receive a win-back offer?
5. How should we evaluate whether our new campaign improved brand engagement?

Exercise 1.10 Find the Flaw in the Output (In-Class Discussion)

1. The denominator problem. A tool calculated conversion rate as conversions divided by impressions, but the class benchmark used conversions divided by clicks. What went wrong at the specify step?
2. The averaged average. A report states that the overall conversion rate is the simple average of the four campaign conversion rates. Use the campaign data from Lab 1.1 to compare this simple average with the pooled conversion rate, calculated as total conversions divided by total clicks. When would the two figures differ most?
3. The causal overclaim. A draft says, “The Loyalty Offer caused the highest ROAS, so StyleCraft should move the full holiday budget to loyalty discounts.” Identify at least three problems.

4. The silent row drop. Cleaning code used df.dropna() and reduced a 10,000-row customer file to 6,200 rows without mentioning it. Why is the output untrustworthy even if the arithmetic is correct?
5. The zero-spend campaign. A campaign with ad_spend = 0 and revenue > 0 produces infinite ROAS. How should the analyst handle this before presenting the table, and which check in Code 1.3 would catch it?
6. The invented metric. A deck reports "incremental reach efficiency" with no formula and no source. Drawing on Section 1.12, explain what you would ask for and what you would do if no formula exists.

Exercise 1.11 Ethics and Privacy Mini-Cases (In-Class Discussion)

For each case, identify the issue, the stakeholders affected, and what a responsible analyst should do instead.

1. The convenient paste. An intern has a CRM export containing customer names, emails, and purchase histories. To save time, the intern pastes 500 rows into a public AI chatbot and asks it to clean and deduplicate the list.
2. The model that learned the past. A response model trained on historical campaign data recommends excluding several ZIP codes because past response rates were low. A colleague notes that StyleCraft has historically spent almost no marketing budget in those areas.
3. The persuasive axis. Preparing an executive slide, an analyst truncates the y-axis of a revenue chart so that a 3 percent increase fills the frame. "It is the same data," the analyst says. "I am just making the story clearer."

Exercise 1.12 From Output to Recommendation (In-Class Discussion)

Read the following output: "The Loyalty Offer has the highest ROAS and the highest post-click conversion rate. The Clearance campaign has the lowest ROAS. The Welcome Series has the lowest revenue and the lowest post-click conversion rate."

1. Write a weak conclusion that overclaims.
2. Write a better interpretation that stays close to the evidence.
3. Write a recommendation that includes one limitation and one next step.

1.15.3 Extensions

Exercise 1.13 Rewrite the Prompt (Optional)

1. Rewrite the prompt "Analyze this campaign data" using the following elements: business question, what one row represents, variables, desired output, constraints, and verification check.
2. Rewrite the prompt "Tell me which campaign is best" using the same elements.
3. Rewrite the prompt "Make a chart for my boss" using the same elements.
4. Rewrite the prompt "Clean this file" using the same elements.
5. Rewrite the prompt "Write a recommendation" using the same elements.
6. After rewriting each prompt, explain what the assistant could still get wrong.

Exercise 1.14 Lab 1.2: Adding Margin (Optional)

ROAS is useful, but it ignores margin. In this lab, make that limitation concrete by adding a gross margin rate to each campaign and recomputing which campaign looks best. In this dataset, margin_rate is the gross margin rate on campaign revenue: the share of revenue remaining after cost of goods sold, before any marketing expenditure. Subtracting advertising spend from gross margin dollars therefore produces net contribution after advertising spend, not contribution margin in the accounting sense; the column is named accordingly so that the two are not confused.

Code 1.4. Add gross margin and compute net contribution

```
campaigns["margin_rate"] = [0.55, 0.32, 0.18, 0.50]

campaigns["gross_margin_dollars"] = (
```

```
    campaigns["revenue"] * campaigns["margin_rate"]
)
campaigns["net_contribution_after_ad_spend"] = (
    campaigns["gross_margin_dollars"]
    - campaigns["ad_spend"]
)

campaigns[[
    "campaign", "roas",
    "net_contribution_after_ad_spend",
]]
```

1. Before running the code, predict whether the net contribution ranking will match the ROAS ranking. Explain why.
2. Run the code. Which campaign generates the highest net contribution in dollars? Which generates the lowest?
3. Hand-check one row and write a two-sentence verification note. For New Arrivals, for example, net contribution after advertising spend is $27,900 × 0.55 − $1,800 = $13,545.
4. Write a four-sentence managerial interpretation that uses both ROAS and net contribution and avoids causal language.
5. A teammate argues that the Clearance campaign should be cut entirely because of its low margin. What business considerations, not visible in this table, might justify keeping it?
6. Explain why net contribution after advertising spend is not the same as contribution margin, and what a finance colleague would need in order to reconcile the two.

Exercise 1.15 Reflection Questions (Optional)

These optional questions connect the chapter to professional practice and your development as an analyst.

1. How do you expect AI tools to change the work of marketing analysts?
2. Which part of the analytics workflow seems most important for preventing mistakes?
3. What concerns do you have about using Python or Google Colab, and how can the workflow used in this guide help you learn responsibly?
4. Think about a brand you know well. What marketing decision could be improved with better data and analysis?
5. What would make a recommendation credible to you if you were the decision-maker?
6. Section 1.12 argued that verification is now part of what an analyst is hired to do. How would you demonstrate that skill in an interview?

1.16 Glossary of Terms

This glossary includes only the terms this chapter introduces. Terms that other chapters own are used with a pointer rather than redefined here; the unit of analysis, data quality, and the formal definition of a metric belong to Chapter 3.

AI in marketing analytics. The use of computational systems to automate tasks, process data, identify patterns, support predictions, personalize actions, or assist communication in marketing research, strategy, and execution (Davenport et al., 2020; Huang & Rust, 2021).

Analytic translation. The process of converting a business problem into an analytical task, interpreting the analytical result in business language, and turning the evidence into a usable recommendation. Course concept for this guide, informed by Provost and Fawcett (2013).

Analytical question. A measurable question that can be addressed with data, comparison, modeling, visualization, or experimentation (Chapman et al., 2000; Provost & Fawcett, 2013).

Business analytics. The use of data, quantitative methods, models, visualization, and technology to understand business problems and support decisions (Camm et al., 2023; Davenport, 2006).

Business question. The managerial issue or decision that an organization needs to address (Provost & Fawcett, 2013; Chapman et al., 2000).

Confabulation. The generation and confident presentation of erroneous content by a generative AI system in response to a prompt (National Institute of Standards and Technology, 2024).

Data-informed decision-making. The use of data and analysis as essential inputs into managerial judgment while recognizing context, ethics, uncertainty, and strategy (Davenport, 2006; Provost & Fawcett, 2013).

Decision context. The managerial situation in which an analysis will be used, including the decision, the stakeholders, the evidence, the constraints, the timing, and the consequences of action. Course concept for this guide.

Descriptive analytics. Analytics that summarizes past or current data to show what happened (Camm et al., 2023; Farris et al., 2015).

Diagnostic analytics. Analytics that investigates why performance may have changed by comparing groups, segments, time periods, or possible drivers, without thereby establishing causality. Course concept for this guide; the comparative methods it relies on are discussed in Camm et al. (2023) and France and Ghose (2019).

Generative artificial intelligence. Systems that produce new content, including text, code, images, audio, or video, in response to a prompt (National Institute of Standards and Technology, 2024).

Input, transformation, and output. The three parts of any analytic step: the data or object used, the calculation or change applied, and the result produced. Course concept for this guide.

Marketing analytics. The application of analytic methods, data, models, and technologies to marketing decisions (Wedel & Kannan, 2016; France & Ghose, 2019).

Metric construction. Every metric is constructed through a definition. That definition specifies what is counted or calculated, the relevant time period, and the records included. For a ratio metric, it also specifies the numerator and denominator.

Predictive analytics. Analytics that uses patterns in existing data to estimate what is likely to happen for new or future observations (Shmueli, 2010; Provost & Fawcett, 2013).

Prescriptive analytics. Analytics that uses evidence, models, constraints, and judgment to recommend actions (Camm et al., 2023; Davenport, 2006).

Responsible analytics. The use of data, methods, AI tools, and recommendations in ways that are accurate, transparent, privacy-aware, fair, and accountable to the decision context (Martin & Murphy, 2017; Barocas & Selbst, 2016; Mittelstadt et al., 2016; Floridi & Taddeo, 2016).

Specify, predict-then-verify, explain, document. The four AI-use habits applied throughout this guide: define the task before asking for help; state an expectation and compare it with the output; be able to explain each step; and record how AI was used and how the work was checked. Author-developed framework for this guide.

Verification. The practice of checking whether analytic output is accurate, reasonable, and aligned with the analytical question before using it in a recommendation (Chapman et al., 2000; Provost & Fawcett, 2013).

1.17 Further Readings

Students who want additional background may begin with the following readings. Marketing analytics sources are listed first because this guide approaches analytics from a marketing decision-making perspective.

- Wedel and Kannan (2016) for a scholarly overview of marketing analytics in data-rich environments.
- France and Ghose (2019) for a broad review of marketing analytics methods and implementation issues.

- Davenport et al. (2020) for an accessible discussion of how AI may change marketing.
- Huang and Rust (2021) for a strategic framework for artificial intelligence in marketing.
- National Institute of Standards and Technology (2024) for an authoritative catalog of generative-AI risks, including confabulation, data privacy, and human oversight.
- Provost and Fawcett (2013) for data analytic thinking and the difference between data and decisions.

1.18 References

Ackoff, R. L. (1989). From data to wisdom. *Journal of Applied Systems Analysis, 16*, 3–9.

Associated Press. (2025, October 7). *Deloitte to partially refund Australian government for report with apparent AI-generated errors*. CTV News. https://www.ctvnews.ca/business/article/deloitte-to-partially-refund-australian-government-for-report-with-apparent-ai-generated-errors/

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732.

Camm, J. D., Cochran, J. J., Fry, M. J., & Ohlmann, J. W. (2023). *Business analytics* (5th ed.). Cengage.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. CRISP-DM Consortium.

Davenport, T. H. (2006). Competing on analytics. *Harvard Business Review, 84*(1), 98–107.

Davenport, T. H., Guha, A., Grewal, D., & Bressgott, T. (2020). How artificial intelligence will change the future of marketing. *Journal of the Academy of Marketing Science, 48*, 24–42. https://doi.org/10.1007/s11747-019-00696-0

Dell'Acqua, F., McFowland, E., III, Mollick, E., Lifshitz, H., Kellogg, K. C., Rajendran, S., Krayer, L., Candelon, F., & Lakhani, K. R. (2026). Navigating the jagged technological frontier: Field experimental evidence of the effects of artificial intelligence on knowledge worker productivity and quality. *Organization Science, 37*(2), 403–423. https://doi.org/10.1287/orsc.2025.21838

Farris, P. W., Bendle, N. T., Pfeifer, P. E., & Reibstein, D. J. (2015). *Marketing metrics: The manager's guide to measuring marketing performance* (3rd ed.). FT Press.

Floridi, L., & Taddeo, M. (2016). What is data ethics? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*(2083), Article 20160360. https://doi.org/10.1098/rsta.2016.0360

France, S. L., & Ghose, S. (2019). Marketing analytics: Methods, practice, implementation, and links to other fields. *Expert Systems with Applications, 119*, 456–475. https://doi.org/10.1016/j.eswa.2018.11.002

Huang, M.-H., & Rust, R. T. (2021). A strategic framework for artificial intelligence in marketing. *Journal of the Academy of Marketing Science, 49*, 30–50. https://doi.org/10.1007/s11747-020-00749-9

Martin, K. D., & Murphy, P. E. (2017). The role of data privacy in marketing. *Journal of the Academy of Marketing Science, 45*, 135–155. https://doi.org/10.1007/s11747-016-0495-4

Mittelstadt, B. D., Allo, P., Taddeo, M., Wachter, S., & Floridi, L. (2016). The ethics of algorithms: Mapping the debate. *Big Data & Society, 3*(2), 1–21. https://doi.org/10.1177/2053951716679679

National Institute of Standards and Technology. (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile* (NIST AI 600-1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.600-1

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big Data, 1*(1), 51–59. https://doi.org/10.1089/big.2013.1508

Rowley, J. (2007). The wisdom hierarchy: Representations of the DIKW hierarchy. *Journal of Information Science, 33*(2), 163–180. https://doi.org/10.1177/0165551506070706

Rust, R. T., Ambler, T., Carpenter, G. S., Kumar, V., & Srivastava, R. K. (2004). Measuring marketing productivity: Current knowledge and future directions. *Journal of Marketing, 68*(4), 76–89. https://doi.org/10.1509/jmkg.68.4.76.42721

Shmueli, G. (2010). To explain or to predict? *Statistical Science, 25*(3), 289–310. https://doi.org/10.1214/10-STS330

Stack Overflow. (2025). *2025 developer survey: AI*. https://survey.stackoverflow.co/2025/ai

Wedel, M., & Kannan, P. K. (2016). Marketing analytics for data-rich environments. *Journal of Marketing, 80*(6), 97–121. https://doi.org/10.1509/jm.15.0413

World Economic Forum. (2025). *The future of jobs report 2025*. https://www.weforum.org/publications/the-future-of-jobs-report-2025/

Chapter 02

Marketing Analytics as Decision Support

Framing Decisions, Stakeholders, and the Analytic Specification

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter presents marketing analytics as a form of decision support. It distinguishes decision support from reporting and from decision automation; describes the anatomy of a marketing decision, including alternatives, stakeholders, constraints, timing, and consequences; and introduces the analytic specification, a short working document that connects a marketing decision to analytical questions, data, candidate methods, deliverables, and a verification plan. Further sections examine how to match analytics types and deliverables to decisions, why baselines discipline interpretation, and why the cost of being wrong is rarely symmetric. A hands-on Google Colab lab turns a vague stakeholder request about StyleCraft's store expansion into a small, verified analysis. The chapter closes with managerial interpretation, industry practice, the ethics of problem framing, and exercises in specification writing and recommendation.

KEYWORDS

decision support; analytic specification; stakeholders; constraints; decision quality; baselines; marketing analytics; data-informed decision-making; artificial intelligence; verification

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Marketing analytics as decision support. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 2, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Tableau is a trademark of Salesforce, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the

underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

1. Explain what it means for analytics to function as decision support, and distinguish decision support from reporting and from decision automation.
2. Describe the anatomy of a marketing decision, including alternatives, stakeholders, constraints, timing, reversibility, and consequences.
3. Identify the stakeholders and constraints that shape an analytics project and explain why they must be identified before analysis begins.
4. Write an analytic specification that connects a marketing decision to analytical questions, a unit of analysis, data, candidate methods, a deliverable, and a verification plan.
5. Match the type of analytics and the form of the deliverable to the decision being supported.
6. Explain why analytic evidence requires a baseline and why the cost of being wrong is usually asymmetric.
7. Use a basic Google Colab workflow to turn a vague stakeholder request into a small, verified analysis and a qualified interpretation.

Chapter 1 introduced the role of the AI-augmented marketing analyst and the workflow that governs this guide: specify, predict-then-verify, explain, and document. This chapter examines the first and most consequential part of that workflow in depth. Before any data is loaded, any code is written, or any chart is drawn, the analyst must understand the decision the analysis is meant to support. The purpose of this chapter is to show how marketing analytics functions as decision support, and to introduce the working document that makes decision support disciplined: the analytic specification.

> **CONCEPT**
>
> **What This Chapter Is Really About**
>
> Chapter 1 argued that analytics becomes valuable when it improves decisions. This chapter asks the practical question that follows from that claim: How does an analyst make sure that an analysis is actually connected to a decision? The answer developed here is that the connection must be designed, not assumed. The analyst identifies the decision, the decision-maker, the stakeholders, and the constraints, and then writes them down in an analytic specification before any code is run. In other words, the most important work in an analytics project often happens before the dataset is opened.

2.1 Marketing Decision Context: Three Requests and One Decision

Chapter 1 left the StyleCraft Collective marketing team at the end of a Monday morning meeting that had stalled. Everyone had data; not everyone had the same question. Two days later, the situation becomes more concrete. Three requests arrive in the analytics team's queue, all within the same afternoon.

The first request comes from the vice president of retail. It reads: "Finance keeps questioning the suburban and resort stores. Can you build us a dashboard showing that the new stores are working? We need it before the quarterly review."

The second request comes from the chief of staff to the chief marketing officer. It reads: "The CFO has asked whether we should pause the remaining store openings planned for next year. The CMO wants a recommendation by the 15th. What would it take to answer this properly?"

The third request comes from the CRM manager. It reads: "Quick one. Can you pull repeat purchase rates by store? No rush."

On the surface, these are three different tasks: a dashboard, a recommendation, and a data pull. Underneath, however, they appear to concern the same business issue: whether StyleCraft's expansion into affluent suburban and resort markets is succeeding. Yet the three requests frame that issue in very different ways. The first request presumes the answer; the analyst is asked to show that the stores are working, not to find out whether they are. The second request identifies a genuine decision, a decision-maker, and a deadline, but leaves the analytical work undefined. The third request specifies a metric but attaches it to no decision at all.

This situation is representative of professional practice. Analysts rarely receive well-formed analytical questions. They receive requests, and requests arrive shaped by the interests, assumptions, and vocabulary of the people who send them. The analyst's first task is therefore not technical. It is interpretive: to find the decision behind the request, and to negotiate a framing of the problem that the evidence can honestly serve.

In this chapter, these three requests will be used as a running example. The chapter will return to them after introducing the concepts needed to handle them well.

> **CONCEPT**
>
> **Requests Versus Decisions**
>
> A request is what a stakeholder asks for: a dashboard, a report, a number, a chart. A decision is the choice the organization must make: continue, pause, expand, reallocate, redesign. Requests are the raw material of analytics work, but decisions are its purpose. When a request arrives without a decision attached, the analyst should ask what decision the output will inform. When a request arrives with the answer already attached, the analyst should ask whether the organization wants advocacy or evidence. This is an operational course concept rather than a standalone academic construct.

2.1.1 Opening Case Questions

1. What decision, if any, sits behind each of the three requests?
2. Which request would you handle first, and why?
3. The first request asks the analyst to show that the stores are working. What risks does this framing create for the analyst, the vice president, and the company?
4. What additional information would you need before agreeing to deliver the CMO's recommendation by the 15th?
5. The third request has no decision attached. Is it therefore harmless? What could go wrong if the analyst simply sends the numbers?

2.2 Reporting, Decision Support, and Decision Automation

The previous section presented three requests that concern the same underlying decision. This section introduces the vocabulary needed to classify what those requests are actually asking analytics to do. Analytics can relate to decisions in three broad ways: it can report on the business, it can support a human decision, or it can automate a decision. The three roles are related, but they place very different demands on the analyst.

Reporting describes the state of the business on a regular cadence. Dashboards, weekly campaign summaries, and monthly revenue reviews are reporting. Reporting is necessary, and much of it is descriptive analytics in the sense defined in Chapter 1 (Section 1.4). However, reporting is not automatically decision support. A dashboard that no decision depends on is monitoring, not support. Organizations can invest heavily in reporting without embedding the resulting insights into a specific decision process, which is one reason analytics investments often fail to change what the organization actually does (LaValle et al., 2011; Sharma et al., 2014).

Decision support connects evidence to a specific choice that a specific person or group must make. The analysis exists because a decision exists. The idea has a long lineage in information systems research, where decision support systems were defined from the start as systems that support, rather than replace, managerial judgment in semi-

structured problems (Gorry & Scott Morton, 1971; Keen & Scott Morton, 1978). Davenport (2006) describes organizations that compete on analytics as those in which analysis is tied to distinctive capabilities and managerial action rather than produced for its own sake. In decision support, the human decision-maker remains responsible for the choice; the analysis narrows uncertainty, clarifies trade-offs, and makes the reasoning inspectable.

Decision automation removes the human from individual choices. A rule or model decides directly: a bidding algorithm sets the price of each ad impression, a send-time model chooses when each email is delivered, a recommendation engine selects which products appear on each customer's home page. Automation becomes a stronger candidate when decisions are repeated at scale, governed by a clear objective, supported by reliable data, rapidly measurable, and bounded by monitoring, guardrails, and a human override. Davenport et al. (2020) distinguish well-defined process automation from context-dependent managerial work and argue that AI is often most valuable when it augments managers, while Huang and Rust (2021) locate repetitive and routine functions in the mechanical domain of AI. Deciding whether to pause a multi-year store expansion strategy has none of those properties. It is infrequent, high-stakes, slow to evaluate, and expensive to reverse. In other words, the StyleCraft expansion question is a decision-support problem, not an automation problem.

Table 2.1

Three Roles of Analytics in Marketing Organizations

Role	Question answered	Typical outputs	Marketing examples	Who decides
Reporting	How is the business doing?	Dashboards, scheduled reports, KPI reviews	Weekly campaign dashboard; monthly revenue by channel	May have no single decision attached; becomes decision support when tied to a defined review and response
Decision support	What should we do about a specific choice?	Analyses, memos, model results, experiment readouts	Whether to pause store expansion; how to reallocate holiday budget	A human decision-maker, informed by evidence
Decision automation	Which action should the system take in each case?	Rules, scores, and models embedded in operations	Real-time bidding; send-time optimization; product recommendations	A system, within limits set by humans

The boundary between the first two roles is not fixed. The same dashboard can be reporting for one audience and decision support for another. The difference is not the visual format; it is whether the output is attached to a choice, an owner, and a defined response. A weekly campaign report is ordinary monitoring when nothing follows from it, and decision support when it carries a review cadence, a threshold, an exception rule, and a named person who acts on it.

DEFINITION

Decision Support

Decision support is the use of data, analysis, models, and visualization to inform a specific managerial choice, while responsibility for the choice remains with a human decision-maker. In this guide, decision support is the default mode of marketing analytics: the analysis exists because a decision exists.

Source: Adapted from Gorry and Scott Morton (1971), Keen and Scott Morton (1978), Davenport (2006), and Sharma et al. (2014).

DEFINITION

Decision Automation

Decision automation is the delegation of individual, repeated decisions to rules, algorithms, or models, within boundaries and monitoring set by humans.

Course heuristic: marketing decisions are stronger candidates for automation when they are repeated at scale, governed by a clear objective, supported by reliable data, rapidly measurable, and bounded by monitoring, guardrails, and a human override. Low reversibility, serious consequences for individual customers, unstable objectives, or unclear accountability argue for human decision support rather than full automation.

Source: Course heuristic developed for this guide, informed by Davenport et al. (2020) and Huang and Rust (2021).

The distinction matters for the three StyleCraft requests. The dashboard request, taken literally, is a reporting task. The CMO's request is a decision-support task. The repeat-rate pull is a fragment that could serve either. The next section examines what a decision-support task requires the analyst to understand: the structure of the decision itself.

2.3 The Anatomy of a Marketing Decision

The previous section distinguished decision support from reporting and automation. This section examines what a decision actually consists of, because an analyst cannot support a decision without being able to describe it. Decision researchers have long argued that decisions are better understood as structured objects than as moments of choice: a decision has alternatives, values at stake, information, and consequences, and the quality of a decision can be assessed by how well those elements were handled (Keeney, 1992; March, 1994).

For marketing analytics, six elements are especially useful to identify.

First, the choice and its alternatives. A decision requires at least two genuine alternatives. "Should we pause the remaining store openings?" implies at least three: continue as planned, pause and reassess, or cancel. If only one alternative is acceptable to the organization, there is no decision to support, and the analyst should recognize the request as advocacy.

Second, the decision-maker. Someone must own the choice. For StyleCraft's expansion question, the CMO and CFO jointly own it; the vice president of retail is an interested party but not the decision-maker.

Third, the stakeholders. These are the people and groups affected by the decision or holding relevant information. Section 2.4 examines them in detail.

Fourth, the constraints. Time, budget, data availability, legal and privacy limits, and organizational realities all bound what analysis is possible. Constraints are examined in Section 2.4 as well.

Fifth, the timing and cadence. Some decisions are one-off and deadline-driven, such as the recommendation due on the 15th. Others recur on a cycle, such as monthly budget reallocation. Recurring decisions justify reusable deliverables such as dashboards and models; one-off decisions usually justify a focused analysis and a memo. This distinction will matter again in Section 2.6.

Sixth, the consequences and reversibility. A steeply discounted flash sale can be evaluated within days and, if it fails, is unlikely to recur. A store lease commits capital for years. The less reversible and more consequential the decision, the more the analysis should invest in verification, in alternative explanations, and in honest statements of uncertainty.

Table 2.2

The Anatomy of a Marketing Decision, Applied to StyleCraft

Element	Question to ask	StyleCraft expansion example
Alternatives	What could the organization actually do?	Continue openings as planned; pause and reassess in two quarters; cancel remaining openings
Decision-maker	Who owns the choice?	CMO and CFO jointly, with board visibility
Stakeholders	Who is affected or holds information?	Retail, finance, CRM, brand, store staff, customers
Constraints	What bounds the analysis and the action?	Recommendation due by the 15th; store-level data only partially centralized; lease commitments already signed for two sites
Timing and cadence	Is this one-off or recurring?	One-off recommendation now; likely to recur as a quarterly review
Consequences and reversibility	How costly is being wrong, and can the action be undone?	Multi-year leases; brand positioning effects; pausing is partially reversible, canceling is not

CONCEPT

Decision Quality Versus Outcome Quality

A good decision can produce a bad outcome, and a bad decision can be rescued by luck. Decision quality refers to how well the choice was framed, informed, and reasoned at the time it was made; outcome quality refers to what later happened. Analysts support decision quality. This distinction protects analysts from two errors: claiming credit when a recommendation happens to work out, and abandoning sound reasoning because a defensible recommendation did not.

Source: Course concept developed for this guide, informed by Keeney (1992), March (1994), and Kahneman et al. (2011).

2.3.1 Levels of Marketing Decisions

It is also useful to classify marketing decisions by their level, a distinction that traces to Anthony's (1965) framework of strategic planning, management control, and operational control. Operational decisions are frequent and narrow: which subject line to send this week, which products to feature in Friday's drop. Tactical decisions allocate resources within an existing strategy: how to split the holiday budget across channels, which segments receive which offers. Strategic decisions change the strategy itself: whether to continue the suburban and resort expansion, whether to reposition the loyalty program. Moving up the levels, decisions become rarer, less reversible, and more dependent on judgment relative to data. Accordingly, the analyst's role shifts from producing repeatable outputs toward framing evidence, surfacing alternative explanations, and communicating uncertainty.

2.3.2 Good Enough, On Time

Finally, decision support is bounded by time. Simon (1997) observed that real decision-makers satisfice: they search for an option that is good enough given limits on time, information, and attention, rather than for a provably optimal one. The same logic applies to the analysis itself. An analysis that would be definitive in six weeks is useless for a recommendation due on the 15th. The professional skill is to design the best analysis that fits the deadline, to state clearly what the compressed analysis cannot establish, and to propose what should be measured next. In other words, "rigorous enough, on time, with stated limits" is usually more valuable than "perfect, late."

2.4 Stakeholders and Constraints

Section 2.3 identified stakeholders and constraints as two elements of every decision. This section examines each in more detail, because both shape the analysis before it begins.

2.4.1 Stakeholders

A stakeholder is any person or group that affects, or is affected by, the decision an analysis supports (Freeman, 1984). Stakeholders matter to the analyst for three practical reasons. First, they hold information: the retail lead knows that two of the newest stores opened with reduced marketing support, a fact that no dataset column records. Second, they hold criteria: finance evaluates the stores on contribution after occupancy costs, while brand evaluates them on positioning in affluent markets; the same evidence will be read differently against different criteria. Third, they hold influence: a technically sound recommendation that ignores a key stakeholder's concern is likely to be contested rather than used.

Furthermore, one stakeholder group is easy to overlook because it is not in the room: customers. Decisions about targeting, discounting, data use, and store closures affect customers directly, and their interests are not automatically represented by any internal function. Section 2.12 returns to this point.

Table 2.3

Common Stakeholders in a Marketing Analytics Project

Stakeholder	Typical criteria	What they contribute	What they may ask of the analyst
CMO / decision-maker	Growth, brand health, defensibility of the choice	The decision itself; strategic context	A clear, qualified recommendation by a deadline
Finance	Margin, contribution, payback	Cost data; scrutiny of assumptions	Evidence that revenue claims survive cost accounting
Retail operations	Store performance, staffing, leases	Operational context; store-level knowledge	Fair treatment of new stores still ramping up
CRM / lifecycle	Retention, repeat purchase, customer value	Customer-level history; program knowledge	Attention to long-term value, not only first purchase
Brand / creative	Positioning, customer perception	Qualitative context	Caution about metrics that ignore brand effects
Legal / privacy	Compliance, data governance	Boundaries on data use	Assurance that customer data is used appropriately
Data engineering	Data availability, definitions, pipelines	Access, documentation, metric definitions	Realistic timelines; consistent metric definitions
Customers (silent stakeholders)	Value, trust, fair treatment	The behavior recorded in the data	Nothing directly; the analyst must represent them

2.4.2 Constraints

A constraint is any condition that bounds what the analysis or the decision can do. Constraints are sometimes treated as obstacles to be lamented. It is more productive to treat them as design inputs: they determine which analytical questions are answerable, which methods are feasible, and which deliverables are appropriate. An analyst who surfaces constraints at the start can design around them; an analyst who discovers them at the end has usually wasted the work.

Table 2.4
Common Constraints on Marketing Analytics Projects

Constraint type	Examples	Implication for the analysis
Time	Recommendation due by the 15th	Choose methods that fit the deadline; state what was not possible
Data	Store-level costs not centralized; short history for new stores	Some questions become unanswerable now; note them as next measurements
Budget and tools	No paid experimentation platform; free-tier Colab	Prefer analyses of existing data over new data collection
Skills	Team new to Python; AI assistance available	Keep methods explainable; verify AI-generated code
Legal and privacy	Customer data cannot leave approved systems	No pasting of customer records into external AI tools (see Section 1.13)
Organizational	The vice president has publicly championed the stores	Framing and communication require care; evidence must be evenhanded

CONCEPT

Constraints as Design Inputs

In this guide, constraints are treated as inputs to analytic design rather than as excuses. The analyst identifies time, data, budget, skill, legal, and organizational constraints before the work begins; chooses questions, methods, and deliverables that fit them; and records in the specification what the constraints made impossible. This record protects the analyst and improves the next iteration of the work.

So far, this chapter has described decisions, stakeholders, and constraints as things the analyst must understand. The next section introduces the document that captures that understanding in writing.

2.5 The Analytic Specification

The previous sections examined the parts of a decision. This section assembles them into the working document at the center of this chapter: the analytic specification. The specification is the written answer to the specify habit introduced in Section 1.7. Chapter 1 applied that habit to a single prompt; this chapter applies it to an entire project.

DEFINITION

Analytic Specification

An analytic specification is a short written document, typically one page, that records the decision to be supported, the decision-maker and deadline, the analytical questions, the unit of analysis, the data to be used, the candidate methods, the intended deliverable, the verification plan, the known limitations, and the planned use of AI assistance. It is written before the analysis begins and revised as the work proceeds. The specification is an operational concept for this guide, adapted from the business understanding phase of CRISP-DM.

Source: Adapted from Chapman et al. (2000) and Provost and Fawcett (2013).

The specification serves four purposes. First, it forces the translation from business question to analytical question, in the sense developed in Section 1.5, to happen explicitly and early. Second, it creates alignment: stakeholders can read one page and object to the framing before the work is done rather than after. Third, it disciplines AI use, because the specification supplies the context that turns a vague prompt into a strong one. Fourth, it creates

accountability: when the analysis is finished, the results can be checked against the questions the specification promised to answer.

Table 2.5
The Analytic Specification Template, Applied to the StyleCraft Expansion Question

Component	Question it answers	StyleCraft example (abridged)
Decision	What choice will this analysis inform?	Whether to continue, pause, or cancel the remaining store openings
Decision-maker and deadline	Who decides, and by when?	CMO and CFO; recommendation due the 15th
Analytical questions	What must be measured or compared?	How do stores compare on average order value, repeat purchase rate, and operating-cost-to-revenue ratio by store type? How do the newest stores compare with older stores at the same age?
Unit of analysis	What does one row represent?	One store (the most recent twelve months for stores open at least one year, and the full available period for stores open less than one year)
Data	What data, from where, covering what period?	Store summary extract from finance and CRM, covering the most recent twelve months for stores open at least one year, and the full available period for stores open less than one year
Candidate methods	What analysis fits the questions and the deadline?	Descriptive comparison by store type; cohort comparison by store age
Deliverable	What will the decision-maker receive?	A two-page memo with one summary table and a stated recommendation
Verification plan	How will the output be checked?	Row counts against the store roster; rate ranges between 0 and 1; totals reconciled to finance figures; one row hand-checked
Limitations and risks	What can this analysis not establish?	No causal claims; the reporting window is not equal across stores, so the newest store is not directly comparable; store costs exclude product cost
AI-use plan	Where will AI assist, and how will it be checked?	Draft pandas code and memo language; all code verified per Section 1.7; documented per Appendix D

Two features of the template deserve emphasis. The verification plan is written before the results exist, which is the predict-then-verify habit applied at project scale: the analyst commits in advance to what "checked" will mean. Moreover, the limitations component is written at the start, not retrofitted at the end. An analyst who knows from the outset that the data cannot support a causal claim is far less likely to draft one under deadline pressure.

2.5.1 Example: Rewriting the Dashboard Request

Consider the first request from Section 2.1: "Can you build us a dashboard showing that the new stores are working?" A specification-minded analyst would respond by proposing a reframing, in roughly the following form: "I understand the quarterly review is the deadline. To be useful there, I would propose an analysis of how the new stores are performing, on metrics finance will accept, with new stores compared against older stores at the same age. If the evidence is favorable, it will be far more persuasive coming from an evenhanded analysis than from a dashboard built to advocate."

Notice what the reframing changes. The presumed conclusion ("the stores are working") becomes an open question. The deliverable is negotiated from a standing dashboard, which suits a recurring decision, to a focused analysis, which suits the one-off review. Furthermore, the stakeholder's legitimate need, being well prepared for the quarterly review, is preserved. Reframing is not refusal; it is the conversion of a request into a decision the evidence can honestly serve.

2.6 Matching Analytics and Deliverables to the Decision

The previous section introduced the specification, which includes a candidate method and a deliverable. This section examines how those two choices should be made. Chapter 1 (Section 1.4) introduced the four types of analytics: descriptive, diagnostic, predictive, and prescriptive. The specification asks the analyst to choose among them deliberately, based on the decision rather than on ambition.

A useful discipline is to begin with the least sophisticated analysis that could inform the decision, and to escalate only when the decision requires it. Many managerial decisions are adequately supported by careful descriptive comparison. The StyleCraft expansion question, for instance, begins descriptively: how do the store types actually compare on the metrics finance cares about? Diagnostic work follows if the comparison raises questions of why. Predictive work is justified when the decision depends on what will happen, such as forecasting whether new stores will reach contribution targets. Prescriptive decisions may require causal evidence from experiments when the organization must choose among actions and the cost of choosing incorrectly is high; Chapter 11 examines experiments and causal evidence directly. It is worth separating the two ideas here. An experiment estimates the causal effect of the alternatives; it does not by itself prescribe the action. The prescription comes from combining the experimental result with objectives, costs, constraints, and judgment. In other words, an experiment is a way of generating evidence, not a fifth type of analytics.

The deliverable should likewise follow the decision, and especially its cadence. One-off decisions are usually best served by a memo or briefing: a short document with the recommendation, the evidence, the limitations, and the next measurement. Recurring decisions justify reusable deliverables: a dashboard for monitoring, a model score refreshed on a schedule, an experiment pipeline. Building a dashboard for a one-off decision wastes effort; writing a one-off memo for a decision that recurs monthly wastes the organization's time twelve times a year.

Table 2.6

Matching the Decision to the Analytics and the Deliverable

Decision situation	Evidence or analytics needed first	Typical deliverable	Cadence
Quarterly review of store performance	Descriptive comparison	Analysis memo with summary table	One-off, likely to recur quarterly
Understanding why repeat purchase declined	Diagnostic (cohorts, segments)	Analysis memo or briefing	One-off
Monitoring campaign health	Descriptive	Dashboard	Recurring (weekly)
Deciding which customers receive a retention offer	Predictive (propensity)	Model score plus targeting rule	Recurring (each campaign)
Choosing between two offer designs	Causal evidence from a randomized experiment	Experiment readout with lift, uncertainty, and a decision recommendation	One-off per test
Setting the holiday budget split	Prescriptive (allocation with judgment)	Recommendation memo with scenarios	Recurring (annual)

CONCEPT

Decision Cadence

Decision cadence is the frequency with which a decision recurs. Cadence drives the choice of deliverable: one-off decisions favor focused analyses and memos, while recurring decisions justify reusable assets such as dashboards, refreshed model scores, and standing experiments. Mismatched cadence is a common and expensive analytics failure. This is an operational course concept rather than a standalone academic construct.

2.7 Evidence, Baselines, and the Cost of Being Wrong

Section 2.6 matched analyses to decisions. This section examines two properties that any decision-supporting evidence must have: a baseline, and an honest accounting of what being wrong would cost.

2.7.1 Compared to What?

A number by itself supports no decision. Suppose the analyst reports that StyleCraft's suburban stores have an average order value near $119. Is that good? The question is unanswerable until a baseline is chosen: compared to the urban stores? To the same stores last year? To the plan finance approved? To what those customers would have spent online anyway? Each baseline answers a different analytical question, and some baselines flatter while others do not. The suburban stores look strong against an average-order-value baseline and weak against a repeat-purchase baseline, which is precisely why the Chapter 1 meeting stalled.

CONCEPT

Baseline

A baseline is the comparison point against which a metric or result is judged: a prior period, a comparable group, a plan or target, a do-nothing alternative, or a stated expectation. Every claim of the form "X is high," "X improved," or "X is working" implicitly depends on a baseline, and the analyst should make that baseline explicit. Choosing the baseline before seeing the results is a form of predict-then-verify.

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013) and Shmueli (2010).

Baselines also discipline the treatment of new stores. Comparing a store seven months after opening against a store in its sixth year answers almost nothing; comparing it against older stores at the same age is a fairer analytical question. The lab in Section 2.9 makes this concrete.

2.7.2 Asymmetric Errors

Every recommendation can be wrong in two directions, and the two directions rarely cost the same. If StyleCraft pauses an expansion that would have succeeded, it loses growth, momentum with landlords, and credibility for the strategy. If it continues an expansion that is failing, it compounds multi-year lease losses. The analysis cannot eliminate either error, but the recommendation should acknowledge which error is more costly and more reversible, because that asymmetry legitimately influences the decision even when the evidence is ambiguous (Kahneman et al., 2011; March, 1994).

The same logic recurs throughout marketing analytics at smaller scale. A retention model that wrongly flags a loyal customer as at-risk costs one unnecessary discount; a model that misses a genuinely departing customer costs the entire future relationship. Chapter 9 formalizes this reasoning as false positives and false negatives; the managerial habit of asking "which mistake hurts more?" is available now, long before the formal machinery.

2.7.3 Is the Analysis Worth Doing?

Finally, evidence earns its cost by improving the decision it informs. Before commissioning the analysis, the analyst should ask whether any plausible result could change the action, its timing, its scale, its safeguards, or the confidence with which it is taken. If no result could affect any of those, the analysis has little decision value. If the store leases for next year are already signed and cannot be exited, an elaborate analysis of whether to sign them has no decision value, however interesting it may be. Conversely, if a modest analysis could move the choice between pause and continue, it is worth doing well. Formally, the value of information is the improvement in expected decision payoff obtained by reducing uncertainty, net of the cost of obtaining that information (Howard, 1966; Keeney, 1992). Practically, the question is the professional defense against producing analytics for its own sake (Pfeffer & Sutton, 2006).

2.8 AI as a Decision-Support Assistant

The previous sections have been about framing. This section asks where AI tools fit into framing work, because the four habits from Section 1.7 apply here just as they apply to code.

AI assistants can be genuinely useful before any data is touched. Given a description of the situation, an assistant can draft a first version of the analytic specification, propose analytical questions the analyst had not considered, enumerate plausible stakeholders and constraints, and suggest baselines. Used this way, the assistant functions as a brainstorming partner that widens the analyst's view of the problem.

However, the risks at the framing stage are distinctive. A drafted specification can be fluent, plausible, and generic: it may propose questions that fit any retailer rather than StyleCraft specifically, or omit the organizational constraint that no dataset records. More subtly, an assistant will usually accept the framing it is given. Asked to "outline an analysis showing the new stores are working," most assistants will comply, producing a professional-looking plan for advocacy. The assistant amplifies the framing; it does not audit it. Because AI outputs are sensitive to the prompt and may produce fluent but generic plans, the analyst should treat AI-generated specifications as drafts to be audited, not as authority. Responsibility for the framing therefore remains entirely with the analyst.

A practical countermeasure is to use the assistant against the framing deliberately. After drafting a specification, ask the assistant to argue the other side: What would a skeptical CFO say about this plan? Which analytical questions are missing? What alternative explanations could produce the same evidence? This use of AI, sometimes described as red-teaming one's own work, converts the tool from an accelerator of the first idea into a test of it.

> **AI IN PRACTICE**
>
> **Ask for the Other Side**
>
> Before finalizing an analytic specification, submit it to an AI assistant with a prompt of the form: "Here is my analysis plan. Act as a skeptical reviewer. Identify the weakest analytical question, one stakeholder I have ignored, one alternative explanation for the result I expect, and one verification check I have not planned." The assistant's answer is not authoritative, but it is cheap, fast, and frequently catches a genuine gap. Document the exchange as described in Appendix D.

2.9 Hands-On Application in Python and Google Colab

The preceding sections were conceptual. This section moves from concepts to practice by carrying one of the StyleCraft requests through a small analysis. As in Chapter 1, the lab uses a compact dataset created directly in Python so that the focus remains on the workflow rather than on file handling. The store figures below are a simplified preview of StyleCraft's full store and transaction data; later chapters load the complete, realistic datasets from the companion repository. The numbers are constructed for instruction and should not be treated as industry benchmarks.

2.9.1 Lab 2.1: From Request to Specification to First Evidence

The scenario is the CMO's request from Section 2.1: a recommendation on the store expansion is due by the 15th. Assume the specification in Table 2.5 has been agreed upon. The first analytical question reads: How do stores compare on average order value, repeat purchase rate, and operating-cost-to-revenue ratio by store type? The unit of analysis is one store, summarized over the most recent twelve months for stores open at least one year, and the full available period for stores open less than one year. That definition is deliberate rather than tidy. Scottsdale has been open seven months, so its row covers seven months and not twelve, and its totals are reported as observed rather than annualized. The windows are therefore not equal across stores, which is a comparability limitation the specification records before any number is produced.

Before writing any code, apply predict-then-verify. The expectations below are written down first, so that the output has something to be checked against.

VERIFICATION CHECK

Before you run: the table should contain seven stores and eight columns. StyleCraft's catalog runs from roughly $24 to $90 a unit and baskets contain one or more items, so average order value should be positive and should plausibly fall below a few hundred dollars. Repeat purchase rate, expressed as a proportion, must fall between 0 and 1. Based on the Chapter 1 meeting, expect the suburban and resort stores to show higher average order values but lower repeat rates than the urban stores.

After you run: the frame should report seven rows and eight columns, no column should contain missing values, and every store should appear exactly once. The metric columns computed in Code 2.3 should then fall inside the ranges predicted above.

Investigate if: a repeat rate falls outside 0 and 1, an average order value falls outside the predicted range, or an operating-cost ratio exceeds 1. Note that the catalog floor is not a floor for the metric, because discounts, returns, credits, and partial refunds can pull a realized average order value below the cheapest list price. A ratio above 1 is not an arithmetic error, but it must be explained before the table is discussed.

Code 2.1. Create the store summary table

```
import pandas as pd

stores = pd.DataFrame({
    "store": ["SoHo", "West Village", "Brooklyn", "Rye",
              "Greenwich", "Hamptons", "Scottsdale"],
    "store_type": ["Urban", "Urban", "Urban", "Suburban",
                   "Suburban", "Resort", "Resort"],
    "months_open": [60, 48, 40, 30, 24, 20, 7],
    "revenue": [3240000, 2150000, 1890000, 1610000,
                1780000, 1420000, 610000],
    "orders": [49850, 33600, 30050, 13650, 14850, 11450, 5050],
    "customers": [20600, 14200, 13900, 9150, 10300, 8900, 4300],
    "repeat_customers": [9680, 6320, 5890, 1690, 1850, 1160, 310],
    "operating_cost": [2050000, 1430000, 1260000, 1330000,
                       1490000, 1260000, 720000]
})

stores
```

Expected output: a table with seven rows, one per store, and eight columns.

Each row represents one store. The columns record the store type, the number of months the store has been open, revenue for the store's reporting window, order and customer counts, the number of customers who purchased

more than once in the window, and the store's operating cost. One definition deserves attention before any metric is computed: operating_cost here includes occupancy, staffing, and local marketing, but excludes the cost of the products sold; the name is chosen deliberately so that no reader mistakes it for total cost. This matters for interpretation later, and noticing it now is part of the specify habit.

2.9.2 Verifying Structure Before Computing Metrics

The first verification checks concern structure, not conclusions.

Code 2.2. Run basic structure checks

```
# Expect 7 rows and 8 columns
stores.shape

# Expect no missing values
stores.isna().sum()

# Expect every store in the roster exactly once
stores["store"].is_unique
```

Expected output: (7, 8); a column of zeros; True.

If the shape, missingness, or uniqueness checks fail, the analysis stops until the discrepancy is explained. In a realistic project, this is also the point at which totals would be reconciled against finance's own figures, as the verification plan in Table 2.5 promised.

2.9.3 Computing the Decision Metrics

The next cell computes the three metrics named in the analytical question, plus revenue per customer as supporting context.

Code 2.3. Calculate store-level decision metrics

```
stores["aov"] = stores["revenue"] / stores["orders"]
stores["repeat_rate"] = stores["repeat_customers"] / stores["customers"]
stores["operating_cost_ratio"] = stores["operating_cost"] / stores["revenue"]
stores["revenue_per_customer"] = stores["revenue"] / stores["customers"]

stores[["store", "store_type", "months_open",
        "aov", "repeat_rate", "operating_cost_ratio"]].round(3)
```

Expected output (selected rows): SoHo shows aov 64.995, repeat_rate 0.470, operating_cost_ratio 0.633; Rye shows 117.949, 0.185, 0.826; Scottsdale shows 120.792, 0.072, 1.180.

Now compare the output with the predictions. Average order values run from roughly $63 to $65 for the urban stores and from roughly $118 to $124 for the suburban and resort stores, all positive and all well below the predicted ceiling. Repeat rates run from about 0.42 to 0.47 in urban stores, about 0.18 in suburban stores, and lower still in the resort stores, all between 0 and 1. The direction matches the expectation from the Chapter 1 meeting. One value stands out: Scottsdale's operating-cost ratio exceeds 1, meaning the store's operating cost exceeded its revenue in the window. A ratio above 1 is not an arithmetic error, but it is exactly the kind of value that must be flagged, because it will dominate any discussion of the table.

2.9.4 Summarizing by Store Type, Carefully

The analytical question asks for comparison by store type. There are two ways to aggregate, and they answer different questions, a distinction first raised in Exercise 1.10, item 2, "The Averaged Average."

Code 2.4. Compare equal-weighted and pooled aggregation

```
# Equal-weighted average of store-level rates (each store counts equally)
stores.groupby("store_type")[["aov", "repeat_rate",
                              "operating_cost_ratio"]].mean().round(3)

# Weighted (pooled) rates built from the underlying totals
totals = stores.groupby("store_type")[["revenue", "orders",
                                       "customers", "repeat_customers",
                                       "operating_cost"]].sum()
totals["aov"] = totals["revenue"] / totals["orders"]
totals["repeat_rate"] = totals["repeat_customers"] / totals["customers"]
totals["operating_cost_ratio"] = totals["operating_cost"] / totals["revenue"]
totals[["aov", "repeat_rate", "operating_cost_ratio"]].round(3)
```

Expected output, pooled version: Urban 64.141, 0.449, 0.651; Suburban 118.947, 0.182, 0.832; Resort 123.030, 0.111, 0.975. The equal-weighted averages differ slightly, for example Resort 122.405, 0.101, 1.034.

The equal-weighted mean answers a per-store question: what does the average store of this type look like when every store receives the same weight? The pooled version answers a portfolio question: what is the aggregate rate for this store type when each metric is reconstructed from its underlying totals? Note that each pooled metric is weighted by its own denominator: average order value by orders, repeat rate by customers, and the operating-cost ratio by revenue. The habit to build is therefore not only "which denominator?" but "which unit receives the weight?" — the store, the order, the customer, or the revenue dollar. Chapter 3 returns to that question when it defines grain and the elements of a metric formally. With these seven stores the two versions differ only modestly, for example a resort repeat rate of about 0.10 under the equal-weighted mean versus about 0.11 pooled, but the gap widens whenever group members differ greatly in size. For a finance audience evaluating the economics of the expansion, the pooled version is usually the right default.

The pooled results tell a compact story. Urban stores show an average order value near $64, a repeat rate near 0.45, and an operating-cost ratio near 0.65. Suburban stores show an average order value near $119, a repeat rate near 0.18, and an operating-cost ratio near 0.83. Resort stores show the highest average order value, near $123, the weakest repeat rate, near 0.11, and an operating-cost ratio near 0.98, meaning operating costs consume nearly all revenue before the cost of the products themselves is even counted. Section 2.10 asks what may, and may not, be concluded from this.

2.9.5 One More Verification: Hand-Check a Row

The final planned check is a hand computation. For Rye: 1,610,000 divided by 13,650 orders is approximately $117.9 of average order value; 1,690 repeat customers divided by 9,150 customers is approximately 0.185; 1,330,000 divided by 1,610,000 is approximately 0.826. All three match the computed table. A single hand-checked row does not prove the table correct, but it catches reversed numerators, wrong columns, and unit errors, which are among the most common failures in practice, and it costs one minute.

2.10 Marketing Interpretation and Managerial Insight

The previous section produced a verified table. This section asks what the table means for the decision, which is where decision support succeeds or fails.

A weak interpretation would say: "The new stores are losing money and customers do not come back. StyleCraft should cancel the expansion." This reading overreaches in at least three ways. First, it treats the resort operating-cost ratios as a settled verdict, when Scottsdale has been open seven months: a store that new has had little time to accumulate repeat customers, and its costs include one-time opening expenses. Its repeat rate of about 0.07 is partly a mechanical consequence of its age, which is why the specification's second analytical question, comparing new stores against older stores at the same age, exists. Second, the table says nothing about why suburban customers do not return: the data cannot distinguish a customer who defected from one who shifted to the online channel, a distinction the current unit of analysis, the store, cannot see. Third, no causal claim is available: the stores differ in

age, location, customer mix, and marketing support, so the comparison describes differences without explaining them.

A stronger interpretation would say: “Across each store’s reporting window, the suburban and resort stores generate substantially higher average order values than the urban stores, roughly $119 to $123 versus $64, but convert far fewer customers into repeat buyers, roughly 0.11 to 0.18 versus 0.45, and operate at much heavier operating-cost ratios, roughly 0.83 to 0.98 versus 0.65, before product costs. The newest store’s figures are not yet comparable because of its short history. The evidence is consistent with the concern that the expansion’s economics depend on repeat behavior that has not materialized, but it does not establish why, and it does not yet compare stores at the same age.”

The stronger interpretation is better because it is specific, it is bounded by the unit of analysis, it separates what the evidence shows from what it suggests, and it leaves the decision with the decision-maker while making the decision easier to reason about.

2.10.1 Revisiting the StyleCraft Case

Return, finally, to the three requests from Section 2.1. The CMO’s request is served by the memo the specification promised: the pooled comparison, the qualified interpretation above, a recommendation, and two named next measurements, a same-age store comparison and a customer-level analysis of whether suburban buyers repurchase online. The vice president’s request is served by the reframing in Section 2.5.1; the evenhanded analysis is what makes any favorable evidence credible at the quarterly review. Moreover, the CRM manager’s request can now be answered responsibly: the repeat rates can be sent along with the store-age caveat attached, so that a number pulled “with no rush” does not circulate stripped of the context that makes it interpretable. In other words, one specification has organized three requests around one decision.

The recommendation itself might reasonably be: pause the two most discretionary openings pending the same-age comparison, continue the committed sites, and decide again next quarter with the two named analyses in hand. Whether the CMO accepts that recommendation depends on judgment, risk appetite, and considerations beyond the table, which is exactly how decision support is supposed to work.

2.11 Business Analytics in Practice

The previous section interpreted a small table for a fictional retailer. This section leaves StyleCraft behind and looks at how the framing habits developed in this chapter appear in real organizations: where the specification actually lives in a company’s weekly rhythm, what the right deliverable is when the deadline is measured in hours, and where the boundary between decision support and decision automation falls in practice.

2.11.1 Where the Specification Lives

Most analytics requests do not arrive by email. In retail, a great many of them arrive in the weekly trading meeting, a standing review in which merchandising, planning, marketing, e-commerce, and finance work through the trading week together: what sold, what did not, what is on markdown, what is short of stock, and what the coming week’s plan should be. The meeting exists because retail decisions are made on a weekly cadence, and it functions in practice as a decision queue. Someone observes that a category is behind plan, someone else offers an explanation, and within a few minutes an analyst has been asked for something.

What separates teams that use analytics well from teams that merely produce it is what happens in those few minutes. On weaker teams the request leaves the room as a deliverable: send me the outerwear numbers. On stronger teams it leaves as a specification, even an informal one. The analyst asks who owns the decision, what will be decided, and by when. Is the meeting deciding whether to mark down, whether to reorder, or whether to move media budget? Each of those is a different analytical question, with a different deadline and a different acceptable deliverable, and the outerwear numbers serve at most one of them well.

This is not an abstract preference. LaValle et al. (2011) found that the organizations obtaining value from analytics were those that began with business questions rather than with available data, and that embedded the resulting insights into an actual business process. Sharma et al. (2014) argue similarly that the returns to analytics

depend on the organizational decision-making processes surrounding the analysis rather than on the analytical technology itself. A weekly trading meeting is exactly such a process. The specification does not have to be a formal document to do its work; it has to exist before the analyst leaves the room. The practice lesson is that in most organizations there is a recurring meeting where decisions are actually made, and locating it is among the highest-return things a new analyst can do.

2.11.2 Forty-Eight Hours, and the Right Answer Was a Memo

Not every decision leaves room for the analysis an analyst would prefer. A common situation in marketing is the mid-quarter reallocation: a channel is underdelivering, a competitor has moved, or a budget line must be committed before the end of the week, and the analytics team has two days. Two days is not enough to build a marketing mix model or to run an incrementality test. It is enough to assemble what is already known, state the assumption behind each number, and write down what the evidence can and cannot support.

Under that constraint the strongest deliverable is usually a memo rather than a model. That is a less modest claim than it sounds. Amazon, a company with no shortage of modeling capacity, does not use slide decks for decisions of consequence. “We don’t do PowerPoint (or any other slide-oriented) presentations at Amazon,” Bezos wrote in the 2017 letter to shareholders. “Instead, we write narratively structured six-page memos. We silently read one at the beginning of each meeting in a kind of study hall” (Bezos, 2018). The reasoning is that prose forces the writer to make the argument explicit, including the parts that a bulleted chart can leave comfortably ambiguous.

For a forty-eight-hour reallocation request, a memo does several things a rushed model cannot. It states the decision and the alternatives. It presents the evidence with its baseline attached. It says plainly which comparisons are confounded and which figures are estimates. And it names the measurement that would settle the question given more time, which converts an admitted limitation into a proposal. Simon’s (1997) account of satisficing applies directly: the decision-maker is not searching for the optimal answer but for the best defensible one available before the money is committed. The practice lesson is that matching the deliverable to the deadline is an analytic decision rather than an administrative one. A model delivered after the budget is committed is worth nothing, however good it is.

2.11.3 The Automation Boundary

Section 2.2 distinguished decision support from decision automation. In practice that boundary is drawn far more sharply than most conversations about AI suggest, and marketing offers an unusually clean pair of examples.

At one end is search advertising. Google’s automated bidding is explicit about what it does: it sets bids for each individual auction rather than a few times a day, using signals available at auction time, including device, location, time of day, audience list, and query text, together with combinations of those signals (Google, n.d.). No human could participate at that frequency. The objective is stated numerically, each auction is individually inexpensive, feedback arrives within hours, and a bad setting can be changed immediately. Every condition in this chapter’s automation heuristic is satisfied. The analyst’s job here is not to set bids. It is to define the conversion the system optimizes toward, to watch the guardrails, and to notice when the objective the system is maximizing has stopped matching the objective the business cares about.

At the other end is brand repositioning. In January 2009, Tropicana relaunched its Pure Premium line with a redesigned carton that replaced the familiar orange-and-straw image. Sales of the line fell roughly 20 percent against the same period a year earlier, the company reverted to the previous packaging by the end of February, and a subsequent demand study estimated the cost of the episode at approximately $27 million (Lee et al., 2010). That decision was infrequent, expensive, slow to evaluate, and, once shelf presence and customer perception had moved, only partly reversible. It also depended on something no dataset contained: how much of the brand’s value sat in a picture customers used to find the product.

Nothing in the automation heuristic would have permitted delegating that choice to a model, and nothing in the bidding case would have justified deciding each auction by hand. The two examples differ not in analytical sophistication but in the structure of the decision. Davenport et al. (2020) draw the same line between well-defined process automation and context-dependent managerial work, and Huang and Rust (2021) locate repetitive, routine functions in the mechanical domain of AI while reserving judgment-dependent work for people. The practice lesson

is that the question is never whether AI is capable. It is whether the decision has the structure that makes delegation responsible.

2.11.4 In Your First Analyst Job

Across these three situations, one failure recurs more often than any technical error: answering the wrong question well. It is a comfortable failure, because everything about the work looks correct. The code runs, the chart is clean, the totals reconcile, and the analyst can defend every step. The only defect is that the output does not bear on the choice the organization is making, and that defect is close to invisible from inside the analysis.

The protection is unglamorous. Before beginning, find the decision, the person who owns it, and the date by which it must be made, then write those three things down where the stakeholder can see them and disagree. Ask what the deliverable will be used for, not only what it should contain. When the deadline is too short for the analysis you would prefer, say what you can support and what you cannot, and propose the measurement that would close the gap.

In your first analyst job, no one will hand you the specification. Requests will arrive as requests, in meetings, in messages, and in hallways, and they will sound urgent. The habit of converting them into decisions before opening the data is what separates an analyst from a person who produces output.

2.12 Ethics, Framing, and Responsible Decision Support

So far this chapter has treated framing as a matter of effectiveness. This section treats it as a matter of responsibility, extending the discussion of responsible analytics from Section 1.13.

The first issue is predetermined framing. The request to "show that the stores are working" asks the analyst to begin from the conclusion. Producing advocacy dressed as analysis misleads the decision-maker, exposes the organization to a worse decision, and spends the analyst's credibility, which is the professional asset everything else depends on. The responsible response, illustrated in Section 2.5.1, is to reframe toward an open question while honoring the stakeholder's legitimate need.

The second issue is metric gaming. When a measure becomes a target, behavior reorganizes around the measure and the measure degrades as evidence (Campbell, 1979; Strathern, 1997). If store managers learn that repeat rate decides the expansion's fate, enrollment tactics that inflate measured repeat purchases without creating real loyalty become tempting. Analysts should anticipate this: prefer a small set of complementary metrics over a single target, and treat sudden improvement in a targeted metric as a finding to investigate rather than a success to announce.

The third issue is who the framing serves. Every stakeholder in Table 2.3 is represented in the room except the customers whose behavior constitutes the data. Decisions framed purely on contribution can quietly disadvantage groups of customers, for example by withdrawing stores, service, or offers from the markets where the company has historically invested least, an echo of the disparate-impact concern raised in Section 1.13 (Barocas & Selbst, 2016). Privacy obligations likewise do not appear in any metric, yet they bind the analysis throughout (Martin & Murphy, 2017). A responsible specification names these considerations explicitly, so that they are weighed rather than omitted.

> **CONCEPT**
>
> **Responsible Framing**
>
> Responsible framing is the practice of posing analytical questions as open questions rather than predetermined conclusions, choosing metrics with awareness of how they may be gamed, and representing affected parties, including customers, who are not present when the decision is framed. It extends responsible analytics (Section 1.13) from the use of data to the design of the question.
>
> *Source: Course concept developed for this guide, informed by Campbell (1979), Strathern (1997), Barocas and Selbst (2016), and Martin and Murphy (2017).*

2.13 Chapter Summary

This chapter developed the claim, introduced in Chapter 1, that analytics becomes valuable when it improves decisions. The main point is that the connection between analysis and decision must be designed. Analytics can report, support, or automate; this guide's default mode is decision support, in which evidence informs a choice that remains with a human decision-maker. A decision has an anatomy, alternatives, a decision-maker, stakeholders, constraints, a cadence, and consequences, and the analyst must be able to describe it before supporting it.

The chapter's central instrument is the analytic specification: one page, written before the work begins, recording the decision, the analytical questions, the unit of analysis, the data, the candidate methods, the deliverable, the verification plan, the limitations, and the AI-use plan. The chapter also argued that every claim needs an explicit baseline, that the two ways of being wrong rarely cost the same, and that an analysis is worth doing only if some result could change the decision. The lab practiced the full arc on StyleCraft's store expansion: request, specification, verified evidence, qualified interpretation, and recommendation. The chapter then stepped outside StyleCraft to see where the specification lives in a real weekly trading meeting, why a forty-eight-hour budget request is often best answered with a memo rather than a model, and where the boundary between automated bidding and brand repositioning falls.

Looking ahead, the specification requires the analyst to name a unit of analysis, variables, and metrics, and to trust the measurements behind them. The next chapter examines exactly those foundations: data, measurement, and marketing variables, including metrics, KPIs, dimensions, features, validity, reliability, and the data dictionary that makes a dataset usable by a team.

2.14 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: find the decision behind the request, describe its anatomy, identify stakeholders and constraints, write the specification, choose the baseline, and connect verified evidence to a qualified recommendation. They are organized into three groups. Core chapter practice is the required path and should be completed by every student. In-class activities are designed for discussion. Extensions are optional and go beyond the required material. Each exercise also carries its assignment label (required practice, in-class discussion, homework submission, or optional) so that instructors can assign selectively.

2.14.1 Core Chapter Practice

Exercise 2.1 Concept Check (Required Practice)

1. Explain the difference among reporting, decision support, and decision automation, with one marketing example of each.
2. What is the difference between a request and a decision? Why does the difference matter?
3. List the components of an analytic specification and explain why the verification plan is written before the results exist.
4. Why does every claim of the form "the metric improved" depend on a baseline?
5. What is the difference between decision quality and outcome quality, and why does the distinction protect analysts?
6. Why is deciding whether to pause a store expansion a poor candidate for decision automation?

Exercise 2.2 Classify the Decision (Required Practice)

For each situation, classify the decision as operational, tactical, or strategic; state whether it is one-off or recurring; and name the deliverable you would propose.

1. Choosing the subject line for this week's new-arrivals email.
2. Setting the split of the holiday budget across paid social, paid search, and influencer spend.
3. Deciding whether to reposition the loyalty program around experiences rather than discounts.
4. Selecting which lapsed customers receive next month's win-back offer.
5. Monitoring whether campaign conversion rates are drifting downward.

6. Deciding whether to open a second resort store in Florida.
7. Choosing which products appear in Friday's drop.
8. Reviewing store performance for the quarterly business review.

Exercise 2.3 Identify Stakeholders and Constraints (Required Practice)

For each situation, name at least three stakeholders, the criteria each would apply, and two constraints the analyst should surface before starting.

1. The CMO asks whether StyleCraft should shift budget from influencer marketing to paid search.
2. Finance asks whether the loyalty program's discounts are eroding margin.
3. The retail lead asks for an analysis of staffing levels against store traffic.
4. Legal asks whether a proposed lookalike-audience campaign uses customer data appropriately.
5. The CRM manager asks which customers should be excluded from discount campaigns.

Exercise 2.4 Write the Analytic Specification (Homework Submission)

Choose the second or third request from Section 2.1, or a request supplied by your instructor. Using the template in Table 2.5, write a one-page analytic specification. Every component must be completed, including the verification plan, the limitations, and the AI-use plan. Then write one paragraph explaining which component was hardest to complete and why.

Exercise 2.5 Choose the Baseline (Required Practice)

For each claim, propose two different baselines, state how each baseline changes the analytical question, and identify which baseline a skeptical CFO would prefer.

1. "The Greenwich store had a strong year."
2. "Repeat purchase improved after the loyalty redesign."
3. "Our TikTok acquisition costs are low."
4. "The Scottsdale store is underperforming."
5. "Email is our best channel."

Exercise 2.6 Hands-On Colab Practice (Homework Submission)

1. Open a new Google Colab notebook and reproduce Lab 2.1.
2. Add a column called revenue_less_operating_cost that subtracts operating_cost from revenue. Because product cost, corporate overhead, and taxes are excluded, this figure is a partial operating comparison rather than contribution or profit, and the column name should say so. Before running it, predict which store type will rank last.
3. Recompute the pooled store-type summary including revenue_less_operating_cost. Does the ranking match your prediction?
4. Compute the simple and pooled repeat rates for the resort stores and report both. Explain in two sentences why they differ and which one you would show a finance audience.
5. Remove Scottsdale from the table and recompute the resort summary. Write two sentences on how much the newest store was driving the resort figures, and what that implies for the same-age comparison proposed in the specification.
6. Write one verification check you performed and one limitation of this dataset for the expansion decision.

Exercise 2.7 AI-Assisted Practice (Homework Submission)

1. Submit your Exercise 2.4 specification to an AI assistant using the red-teaming prompt from Section 2.8. Record what the assistant identified.
2. Revise the specification in response to at most two of the assistant's points, and reject at least one point with a one-sentence justification.
3. Document the exchange using the full AI-use documentation template in Appendix D.

Exercise 2.8 Managerial Memo Exercise (Homework Submission)

Write a memo of 250–350 words from the analytics team to the StyleCraft CMO and CFO, dated before the 15th, recommending a course of action on the remaining store openings. Use the Lab 2.1 evidence. The memo must state the decision, the recommendation, the evidence with its baseline made explicit, one limitation, the asymmetry between the two ways of being wrong, and two named next measurements. Recommended structure: decision context, evidence, recommendation, limitations and risks, next steps.

2.14.2 In-Class Activities

Exercise 2.9 Find the Flaw in the Framing (In-Class Discussion)

1. The missing decision. A team spends three weeks building a beautiful customer dashboard. At the demo, an executive asks, "So what should we do differently?" and no one can answer. Where in this chapter's terms did the project go wrong?
2. The convenient baseline. An analysis reports that the resort stores' revenue grew 40 percent year over year, without mentioning that the prior year included only a partial season after opening. Which baseline was chosen, and which was avoided?
3. The single metric. Leadership announces that store bonuses will now depend on repeat purchase rate. Predict two behaviors this may produce, and explain the risk in this chapter's terms.
4. The wrong cadence. An analyst answers the monthly budget-allocation question with a bespoke forty-page analysis every month. What is the mismatch, and what should be built instead?
5. The unanswerable question. The CMO asks whether the Hamptons store "would have succeeded" if it had opened two years earlier. What can and cannot be said, and what would you offer instead?
6. The overreached memo. A draft memo states: "The data proves that suburban customers are disloyal." Rewrite the sentence so that it stays within what the Lab 2.1 evidence supports.

Exercise 2.10 Ethics and Framing Mini-Cases (In-Class Discussion)

For each case, identify the issue, the stakeholders affected, and what a responsible analyst should do instead.

1. The friendly deadline. A stakeholder offers to extend the analyst's deadline by a week "if the numbers can come out looking a bit stronger."
2. The taught metric. After repeat rate becomes the expansion's headline metric, one store begins enrolling every walk-in customer in the loyalty program at checkout and logging a token second "purchase" through free-gift redemptions.
3. The quiet exit. A contribution-ranked analysis recommends closing the two stores in the markets where StyleCraft has historically spent the least on marketing. No one in the meeting represents the customers in those markets.

2.14.3 Extensions

Exercise 2.11 Rewrite the Request (Homework Submission or Optional Extension)

Rewrite each request below as an open analytical framing. For each, name the decision, the decision-maker you would confirm, and the deliverable you would propose.

1. "Build a dashboard proving that influencer marketing drives our growth."
2. "Get me a number that shows the app is worth the investment."
3. "Pull whatever data supports raising the free-shipping threshold."
4. "Show the board that Gen Z loves the brand."
5. "Just send me last month's numbers."

Exercise 2.12 Reflection Questions (Optional)

Finally, the following questions are intended to be thought-provoking rather than graded.

1. Think of an organization you know. Can you identify one report it produces that no decision depends on? What decision could it be attached to?
2. Have you ever been asked, in any setting, to find evidence for a conclusion that was already chosen? How did you, or how would you now, respond?
3. Which component of the analytic specification do you expect to be hardest to complete in real projects, and why?
4. When, in your judgment, should a marketing decision be automated, and what would make you comfortable delegating it?
5. What would make you trust a recommendation enough to act on it if you were the CMO?

2.15 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Analytic specification. A short written document, prepared before analysis begins, that records the decision to be supported, the decision-maker and deadline, the analytical questions, the unit of analysis, the data, the candidate methods, the deliverable, the verification plan, the limitations, and the planned use of AI assistance (adapted from Chapman et al., 2000; Provost & Fawcett, 2013).

Baseline. The comparison point against which a metric or result is judged, such as a prior period, a comparable group, a plan, a do-nothing alternative, or a stated expectation (adapted from Provost & Fawcett, 2013; Shmueli, 2010).

Constraint. Any condition, including time, data, budget, skills, legal or privacy limits, and organizational realities, that bounds what an analysis or decision can do (operational course concept).

Decision automation. The delegation of individual, repeated decisions to rules, algorithms, or models within boundaries and monitoring set by humans (Davenport et al., 2020; Huang & Rust, 2021).

Decision cadence. The frequency with which a decision recurs; a driver of the appropriate deliverable (operational course concept).

Decision quality. How well a choice was framed, informed, and reasoned at the time it was made, as distinct from the quality of the outcome that followed (Keeney, 1992; Kahneman et al., 2011; March, 1994).

Decision support. The use of data, analysis, models, and visualization to inform a specific managerial choice, while responsibility for the choice remains with a human decision-maker (Gorry & Scott Morton, 1971; Keen & Scott Morton, 1978; Davenport, 2006; Sharma et al., 2014).

Metric gaming. The reorganization of behavior around a measure once it becomes a target, degrading the measure's value as evidence (Campbell, 1979; Strathern, 1997).

Responsible framing. The practice of posing analytical questions as open questions, choosing metrics with awareness of gaming, and representing affected parties who are absent when the decision is framed (adapted from Campbell, 1979; Strathern, 1997; Barocas & Selbst, 2016; Martin & Murphy, 2017).

Satisficing. Searching for an option, or an analysis, that is good enough given limits on time, information, and attention, rather than provably optimal (Simon, 1997).

Stakeholder. Any person or group that affects, or is affected by, the decision an analysis supports (Freeman, 1984).

Value of information. The worth of evidence, judged by its potential to improve the expected quality or value of the decision it informs, net of the cost of obtaining it (Howard, 1966; Keeney, 1992).

2.16 Further Readings

Students who want additional background may begin with the following readings. Decision-focused sources are listed first because this chapter approaches analytics from the decision side.

- Keeney (1992) for value-focused thinking and structuring decisions before analyzing them.

- Kahneman et al. (2011) for a practical checklist for reviewing evidence behind big decisions.
- March (1994) for an accessible treatment of how decisions actually happen in organizations.
- Keen and Scott Morton (1978) for the classic statement of decision support systems as aids to, not replacements for, managerial judgment.
- LaValle et al. (2011) for why analytics investments often fail to reach decisions.
- Davenport et al. (2020) and Huang and Rust (2021) for the augmentation and automation of marketing decisions.

2.17 References

Anthony, R. N. (1965). *Planning and control systems: A framework for analysis*. Harvard University Graduate School of Business Administration.

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732.

Bezos, J. (2018). *2017 letter to shareholders*. Amazon. https://www.aboutamazon.com/news/company-news/2017-letter-to-shareholders

Campbell, D. T. (1979). Assessing the impact of planned social change. *Evaluation and Program Planning, 2*(1), 67–90. https://doi.org/10.1016/0149-7189(79)90048-X

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. CRISP-DM Consortium.

Davenport, T. H. (2006). Competing on analytics. *Harvard Business Review, 84*(1), 98–107.

Davenport, T. H., Guha, A., Grewal, D., & Bressgott, T. (2020). How artificial intelligence will change the future of marketing. *Journal of the Academy of Marketing Science, 48*, 24–42. https://doi.org/10.1007/s11747-019-00696-0

Freeman, R. E. (1984). *Strategic management: A stakeholder approach*. Pitman.

Google. (n.d.). *Google Ads automated bidding*. Google Ads Help. Retrieved July 20, 2026, from https://support.google.com/google-ads/answer/10964872

Gorry, G. A., & Scott Morton, M. S. (1971). A framework for management information systems. *Sloan Management Review, 13*(1), 55–70.

Howard, R. A. (1966). Information value theory. *IEEE Transactions on Systems Science and Cybernetics, 2*(1), 22–26. https://doi.org/10.1109/TSSC.1966.300074

Huang, M.-H., & Rust, R. T. (2021). A strategic framework for artificial intelligence in marketing. *Journal of the Academy of Marketing Science, 49*, 30–50. https://doi.org/10.1007/s11747-020-00749-9

Kahneman, D., Lovallo, D., & Sibony, O. (2011). The big idea: Before you make that big decision… *Harvard Business Review, 89*(6), 50–60.

Keen, P. G. W., & Scott Morton, M. S. (1978). *Decision support systems: An organizational perspective*. Addison-Wesley.

Keeney, R. L. (1992). *Value-focused thinking: A path to creative decisionmaking*. Harvard University Press.

LaValle, S., Lesser, E., Shockley, R., Hopkins, M. S., & Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. *MIT Sloan Management Review, 52*(2), 21–32.

Lee, J.-Y., Gao, Z., & Brown, M. G. (2010). A study of the impact of package changes on orange juice demand. *Journal of Retailing and Consumer Services, 17*(6), 487–491. https://doi.org/10.1016/j.jretconser.2010.08.003

March, J. G. (1994). *A primer on decision making: How decisions happen*. Free Press.

Martin, K. D., & Murphy, P. E. (2017). The role of data privacy in marketing. *Journal of the Academy of Marketing Science, 45*, 135–155. https://doi.org/10.1007/s11747-016-0495-4

Pfeffer, J., & Sutton, R. I. (2006). Evidence-based management. *Harvard Business Review, 84*(1), 62–74.

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big Data, 1*(1), 51–59. https://doi.org/10.1089/big.2013.1508

Sharma, R., Mithas, S., & Kankanhalli, A. (2014). Transforming decision-making processes: A research agenda for understanding the impact of business analytics on organisations. *European Journal of Information Systems, 23*(4), 433–441. https://doi.org/10.1057/ejis.2014.17

Shmueli, G. (2010). To explain or to predict? *Statistical Science, 25*(3), 289–310. https://doi.org/10.1214/10-STS330

Simon, H. A. (1997). *Administrative behavior: A study of decision-making processes in administrative organizations* (4th ed.). Free Press.

Strathern, M. (1997). "Improving ratings": Audit in the British university system. *European Review, 5*(3), 305–321. https://doi.org/10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4

Chapter 03

Data, Measurement, and Marketing Variables

Units of Analysis, Metrics, Validity, and the Data Dictionary

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter examines the measurement foundations of marketing analytics. It presents measurement as the disciplined translation of marketing concepts into recorded data; distinguishes the grain of a table from the unit of analysis of a claim; and classifies variables by level of measurement and by stored data type. The chapter separates four analytic roles — metric, key performance indicator, dimension, and feature — and shows why a ratio metric requires a stated numerator, denominator, window, and filters. Sections on validity, reliability, proxies, and data quality preview the preparation work of Chapter 4. The chapter's central instrument is the data dictionary, which records column and metric definitions so that a dataset can be used consistently by a team. A Google Colab lab reconstructs a StyleCraft dispute in which three teams report three different repeat purchase rates, and resolves it with a written definition.

KEYWORDS

measurement; operational definition; unit of analysis; grain; levels of measurement; metrics; key performance indicators; validity and reliability; data quality; data dictionary

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Data, measurement, and marketing variables. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 3, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Tableau is a trademark of Salesforce, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the

underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

1. Explain how marketing concepts become recorded data through operational definitions, and why every dataset embeds measurement decisions.
2. Distinguish the grain of a table from the unit of analysis of a claim, identify fact and dimension tables, and explain how aggregation changes the claim an analysis can support.
3. Classify marketing variables by level of measurement (nominal, ordinal, interval, ratio) and by stored data type, and explain why the classification governs which summaries, charts, and models are appropriate.
4. Define and distinguish metrics, key performance indicators (KPIs), dimensions, and features, and recognize one business concept appearing in several analytic roles.
5. Specify a ratio metric completely, by naming its numerator, denominator, time window, and filters.
6. Assess the validity and reliability of marketing measures, including proxy measures, and identify common data quality problems.
7. Build a data dictionary that documents columns, metric definitions, owners, and decision uses, and use it to resolve measurement disagreements.
8. Use a basic Google Colab workflow to inspect variable types, verify them against intended measurement levels, and demonstrate how one metric name can produce several defensible numbers.

Chapter 2 introduced the analytic specification, the one-page document that connects a marketing decision to analytical questions, data, methods, and a verification plan. Completing a specification, however, forces the analyst to make commitments that Chapter 2 deferred: naming a unit of analysis, choosing variables, and defining metrics precisely enough that two people computing them would get the same number. This chapter examines those commitments. Its purpose is to show that every number in an analysis is the product of measurement decisions, and to introduce the working document that makes those decisions inspectable and shared: the data dictionary.

> **CONCEPT**
>
> **What This Chapter Is Really About**
>
> Chapter 2 argued that the most important work in an analytics project happens before the dataset is opened. This chapter makes a companion claim: the most dangerous work in an analytics project is the work nobody notices doing. Every time an analyst accepts a column as given, treats a metric name as self-explanatory, or lets a spreadsheet decide what “active customer” means, a measurement decision has been made silently. Silent decisions cannot be verified, defended, or shared. The discipline developed in this chapter is to make measurement decisions explicit — in the unit of analysis, in the variable types, in the metric definitions, and ultimately in a data dictionary the whole team can read. In other words, a number is a claim, and this chapter is about learning to state the claim precisely.

3.1 Marketing Decision Context: One Metric, Three Numbers

Chapter 2 ended with the analytics team’s memo to the StyleCraft CMO and CFO: pause the two most discretionary store openings, continue the committed sites, and decide again next quarter with two named analyses in hand. The memo was well received. It also had an unintended consequence.

Because the memo made repeat purchase rate the headline evidence, everyone at StyleCraft now wants to talk about repeat purchase rate. At the follow-up meeting, three numbers appear on three slides. The vice president of

retail opens with a reassuring figure: “Sixty percent of our customers come back. Our customer base is loyal.” The CRM manager follows with the figure from the memo: “Repeat purchase rate is 43 percent, and it has been drifting down.” Finally, the finance analyst shares a dashboard from the business intelligence tool: “Store repeat rate is 20 percent. The retail economics do not work.”

The CFO looks at the three slides and asks the question the whole meeting has been building toward: “Which of these numbers is wrong?”

The uncomfortable answer, which this chapter develops, is that none of them is wrong, and that this is precisely the problem. The retail figure counts a customer as a repeat buyer if they ever purchased twice, computed among customers with at least one completed order in the available company history. The CRM figure counts repeat purchases within the trailing twelve months, among customers who were active in that window. The finance dashboard counts repeat purchases in physical stores only, within the same window, among customers who shopped in a store. Same company, same underlying transactions, three defensible computations — and three numbers ranging from 20 percent to 60 percent, each circulating under the same name.

No one at the meeting can adjudicate among the three slides, because no one ever wrote down what “repeat purchase rate” means at StyleCraft. There is no shared definition of the numerator, the denominator, the time window, or the filters. There is, in other words, no data dictionary.

The meeting does not end with a shrug, and this is what makes the episode a decision rather than an anecdote. The CFO closes it by attaching the disagreement to a name and a date: before the quarterly expansion review on October 15, she and the CMO will approve one official definition of repeat purchase rate, and the analytics team is asked to bring the alternatives, a recommendation, and the reasoning behind it.

Three consequences follow from whichever definition is approved. First, it becomes the baseline on the expansion dashboard, the figure against which every subsequent quarter is read. Second, it enters the same-age store comparison that Chapter 2 promised, where it will carry real weight in the decision to continue or cancel the remaining openings. Third — and this is the consequence most easily overlooked — the CRM manager has already been told that her team’s performance target for next year will be expressed in it.

That third consequence converts a documentation task into a managerial one. Chapter 2 observed that once a measure becomes a target, behavior reorganizes around the measure and the measure degrades as evidence (Section 2.12). StyleCraft is about to elevate a phrase that currently denotes three different quantities, and to attach a dashboard, a strategic review, and a team target to whichever one wins. The stake, then, is not accuracy. All three numbers are accurate. The stake is that an organization is about to build three commitments on a word it has never defined.

This situation is representative of professional practice. It is widely observed that measurement disagreements inside organizations are rarely arithmetic disputes; they are definitional disputes wearing arithmetic clothing. The analyst’s task in such moments is not to recompute but to make the definitions visible, and then to help the organization choose the definition that serves the decision. This chapter builds the vocabulary and the tools for doing exactly that, and the chapter’s lab (Section 3.10) reconstructs the three numbers above in miniature so that the mechanism can be verified line by line.

CONCEPT

A Number Is a Claim

Every reported figure is shorthand for a longer sentence of the form: “Applying this definition, to this unit of analysis, over this time window, with these filters, using this data source, yields this value.” When the longer sentence is left unstated, different people silently complete it in different ways, and the same metric name comes to denote different quantities. The discipline of this chapter is to write the longer sentence down.

Source: Course concept developed for this guide, informed by Stevens (1946) and Churchill (1979).

3.1.1 Opening Case Questions

1. Restate each of the three repeat purchase figures as a complete sentence: what is being counted, over what population, in what window, with what filters?
2. Which of the three definitions is most relevant to the store-expansion decision from Chapter 2, and why?
3. The retail figure (60 percent) is the largest and the most reassuring. Is it wrong? Is it misleading? Explain the difference.
4. The approved definition will serve three purposes: a dashboard baseline, a strategic review, and a team performance target. Should one definition serve all three? What would you lose by using different definitions for different purposes?
5. If StyleCraft adopts one official definition, what legitimate needs of the other two teams might be lost? How could those needs still be served?
6. Who in the organization should own the definition of a metric like repeat purchase rate: analytics, the team that uses it most, or the decision-maker it informs?

3.2 From Concept to Column: What Measurement Is

The previous section presented a dispute created by unexamined measurement. This section steps back and asks what measurement is, because the rest of the chapter depends on the answer.

Marketing reasons in concepts: loyalty, engagement, awareness, satisfaction, price sensitivity, brand equity. Datasets contain none of these. Datasets contain columns: `loyalty_tier`, `email_opt_in`, `order_date`, `discount_pct`. Between the concept and the column lies measurement — the set of rules by which observations of the world are assigned numbers, labels, or categories. In a classic formulation, measurement is "the assignment of numerals to objects or events according to rules" (Stevens, 1946, p. 677). The definition is deliberately modest. It reminds the analyst that the numbers were not found in the world; they were produced by rules, and the rules can be good or bad, shared or private, documented or silent.

The rule that connects a concept to a recorded value is called an operational definition. "Repeat customer" is a concept; "a customer with two or more completed orders in the trailing twelve months" is an operational definition. The three slides in Section 3.1 represent three different operational definitions of the same concept, which is why they produce three different numbers without anyone making an arithmetic error. Marketing research has long emphasized that constructs must be carefully translated into measures before they can be analyzed, and that poor translation, rather than poor statistics, is the more common source of misleading findings (Churchill, 1979).

Moreover, most marketing datasets are not designed for analysis at all. They are the by-products of operational systems: the e-commerce platform records orders because it must fulfill them; the email platform records opens because it must render messages; the point-of-sale system records line items because it must charge cards. Each system measures what it needs for its own operation, using its own definitions, identifiers, and time conventions. When the analyst later assembles these by-products into an analytical dataset, the measurement decisions of a dozen systems arrive embedded in the columns, mostly undocumented. This is why the analyst must learn to read a dataset the way an editor reads a manuscript: asking of every column where it came from, what rule produced it, and what it can and cannot support.

DEFINITION

Measurement

Measurement is the assignment of numbers, labels, or categories to observations according to explicit rules. In marketing analytics, measurement translates marketing concepts (such as loyalty or engagement) into recorded variables (such as `loyalty_tier` or `email_opt_in`) through operational definitions.

Source: Adapted from Stevens (1946) and Churchill (1979).

DEFINITION

Operational Definition

An operational definition is the specific, written rule by which a concept is turned into a recorded or computed value, including the population it applies to, the time window, and any filters or exclusions. Two analysts applying the same operational definition to the same data should obtain the same result.

Source: Adapted from Churchill (1979) and Carmines and Zeller (1979).

Two consequences of this view run through the rest of the chapter. First, because measurement is rule-governed, the rules can be written down, and Section 3.8 introduces the document in which teams write them: the data dictionary. Second, because measurement is a translation, the translation can fail: a column can measure its concept poorly (a validity problem, Section 3.6), inconsistently (a reliability problem, Section 3.6), or incorrectly (a data quality problem, Section 3.7). Before examining how measurement fails, however, the next section examines the most fundamental measurement decision of all: what one row of the dataset represents.

3.3 Units of Analysis and the Grain of a Dataset

The previous section described measurement as the translation of concepts into columns. This section examines the decision that comes before any column: what entity each row of a dataset describes, and what entity the analysis is actually making a claim about. In data-management practice the first of these is called the grain of a table — the answer to the question, "one row equals one what?" — and declaring it is treated as the first and most important step in designing a table (Kimball & Ross, 2013). The second is the unit of analysis: the entity about which the analysis makes its claim. Chapter 1 introduced the idea informally (Section 1.8), and the analytic specification in Chapter 2 required it to be named (Table 2.5). This section explains why the requirement exists, and why the two ideas, closely related as they are, are worth keeping apart.

The two usually coincide by the time an analysis is finished, and that is exactly what makes the distinction easy to lose. An analyst asking what share of customers repurchase wants a customer-grain table and will build one, at which point grain and unit of analysis agree. But the agreement is manufactured, not given. The available data may arrive at order-line grain while the claim is about people. The analyst's job is then to move the data from the grain it has to the grain the claim requires, and — this is the part that gets skipped — to know that the move has been made. In other words, grain is a property of a table, unit of analysis is a property of a claim, and the work in between is aggregation.

Consider StyleCraft's data family, which later chapters use throughout. It consists of five base tables, each with a different grain, summarized in Table 3.1. The `transactions` table is StyleCraft's primary sales fact table: one row is one order line, carrying the quantitative measures — quantity, revenue, discount — that the company's sales metrics aggregate. The `customers`, `products`, and `stores` tables are dimension tables: one row describes one entity and supplies the attributes by which the sales facts will later be sliced.

The `campaigns` table is the awkward case, and it is more useful to name the awkwardness than to smooth it over. It supplies descriptive attributes such as channel and campaign name, which makes it behave like a dimension; but it also carries spend, impressions, and clicks, which are additive measures in their own right, recorded at campaign grain. A table of that shape is better described as a campaign-level performance table with campaign attributes than as a pure dimension. Note in particular what it is not: campaign spend and impressions are not aggregations of sales transactions. They are separate facts, produced by a different system at a different grain, and bringing them together with sales is an analytical act rather than a roll-up.

Table 3.1

The StyleCraft base tables and their grain

Table	Grain (one row =)	Role	Example columns
transactions	one order line	Primary sales fact table; carries the quantitative sales measures that sales metrics aggregate	order_date, quantity, unit_price, discount_pct, line_revenue
customers	one customer	Dimension: who they are, how acquired, where, loyalty status	signup_date, acquisition_channel, age_band, loyalty_tier
products	one product (SKU)	Dimension: category, price, cost, newness	category, unit_price, unit_cost, is_occasionwear
stores	one physical store	Dimension: location, type, opening date	store_type, metro, opening_date
campaigns	one marketing campaign	Campaign-level performance table: descriptive attributes plus its own quantitative campaign measures, not an aggregation of sales	channel, spend, impressions, clicks

DEFINITION

Fact Table and Dimension Table

A fact table records events or measurements at a declared grain and typically carries quantitative measures, many of which can be aggregated to construct metrics. A dimension table describes entities and supplies the attributes by which those measures are sliced, grouped, and filtered. A data family may contain more than one fact table: sales facts and campaign performance facts are recorded by different systems at different grains, and neither is derived from the other. A table carrying both entity attributes and its own quantitative measures, such as StyleCraft's `campaigns`, is a performance table rather than a pure dimension.

Source: Adapted from Kimball and Ross (2013).

The grain matters because it determines what questions a table can answer directly. A question about baskets ("what is the average order value?") requires order-level or finer-grained data, or a preaggregated table that already carries the revenue and order totals the metric needs. A question about people ("what share of customers repurchase?") requires customer-level data, or transaction data that can be rolled up to customers. A question about the expansion ("how do store types compare?") requires store-level data, or something that aggregates to it. None of these grains is more correct than the others. Each is an aggregation choice, and each choice discards information: rolling transactions up to customers discards which products were bought; rolling customers up to stores discards which customers did the buying.

3.3.1 Changing the Unit Changes the Claim

Aggregation is not merely a loss of detail; it can change the direction of a finding. Robinson (1950) demonstrated the classic version of the problem: a correlation computed across groups can differ in size, and even in sign, from the correlation computed across the individuals inside those groups. Inferring individual-level relationships from group-level data is known as the ecological fallacy, and marketing analytics offers it constant temptation, because marketing data is so often reported at the group grain — by store, by campaign, by segment, by week.

A StyleCraft example makes the danger concrete. At the store grain, average order value and repeat purchase rate are negatively related: the suburban and resort stores post high average order values and weak repeat rates, while the urban stores show the reverse (Lab 2.1). It would be natural, and wrong, to conclude from this that individual customers who spend more per order are less likely to return. The store-level pattern could be produced

by differences in the customer mix across store types — the occasionwear buyer in Greenwich and the weekly-drop follower in SoHo — rather than by any within-customer trade-off between basket size and loyalty. This chapter cannot settle which explanation is right, and that restraint is the lesson: whether high-basket customers repurchase less is a customer-grain question, and only customer-grain data can answer it. Chapter 5 returns to this comparison explicitly; the point here is that the unit of analysis is part of the claim, not a technical footnote to it.

Therefore, two habits follow. First, every analytical question in a specification should name its unit of analysis, as Table 2.5 requires. Second, every reported number should be traceable to the grain at which it was computed, because a "rate" computed over stores and a "rate" computed over customers are different quantities even when they share a name — a lesson the pooled-versus-simple comparison in Lab 2.1 (Section 2.9.4) has already illustrated from the denominator side.

DEFINITION

Grain and Unit of Analysis

The grain of a table is what one row represents: one order line, one order, one customer, one store, one campaign, one day. In dimensional modeling, declaring the grain is the first design decision a table makes. The unit of analysis of an analytical question is the entity about which the analysis makes its claim.

The two usually align once the data has been prepared, but they are declared separately and can come apart: an analyst may begin with order-line-grain data and make a customer-level claim after aggregation. The obligation is to know, at every step, which one is being talked about.

Source: Adapted from Kimball and Ross (2013) and Robinson (1950).

CONCEPT

The Ecological Fallacy in Marketing

The ecological fallacy is the error of inferring individual-level relationships from group-level data. Marketing dashboards report mostly at group grains (store, campaign, segment, week), so the fallacy is a constant occupational hazard: store-level patterns need not hold customer by customer, and campaign-level patterns need not hold impression by impression. The protection is to match the grain of the data to the grain of the claim.

Source: Adapted from Robinson (1950).

So far this chapter has established what measurement is and what one row represents. The next section classifies what the columns themselves contain.

3.4 Types of Variables and Levels of Measurement

The previous section examined the rows of a dataset. This section examines the columns. A variable is any characteristic recorded for each unit of analysis, and it is widely accepted that not all variables carry the same kind of information. Stevens (1946) proposed the classification that analytics still uses: variables can be measured at the nominal, ordinal, interval, or ratio level, and each level licenses a different set of mathematical operations.

Nominal variables are labels without order. StyleCraft's `acquisition_channel` (TikTok, Instagram, Paid Search, Referral, In-Store, Influencer, Email) names categories, but no channel is "greater than" another. The permissible summaries are counts, proportions, and the mode. Identifiers deserve special mention here: `customer_id` and `order_id` are nominal labels even when they look numeric, and averaging or summing them is meaningless. Treating an identifier as a quantity is one of the most common silent errors in AI-generated analysis code, which is one reason this classification is worth internalizing rather than delegating.

Ordinal variables are ordered categories with undefined distances. StyleCraft's `loyalty_tier` (None < Insider < VIP < Backstage) and `age_band` (18–24 through 55+) carry a genuine order, so medians, percentiles, and comparisons such as "VIP or above" are meaningful. However, the distances between categories are not defined: the "gap" between Insider and VIP is not a quantity, so means and differences are not, strictly speaking, licensed. Marketing data is dense with ordinal variables — tiers, bands, ratings, ranks — and much practical judgment concerns how strictly to honor their ordinal character. Survey research, for example, routinely averages rating scales, a practice that is defensible in some settings and contested in others; the analyst should at minimum know that a choice is being made (Stevens, 1946; Velleman & Wilkinson, 1993).

Interval variables have meaningful distances but no true zero. Calendar dates are the clearest marketing example: the difference between two `order_date` values is a meaningful duration, but the "zero" of a calendar is a convention, so ratios of dates are meaningless — June 2026 is not "twice" March 1013. Usefully, differences of interval variables often become ratio variables: tenure in days, computed as `order_date` minus `signup_date`, has a true zero and supports full arithmetic.

Ratio variables have meaningful distances and a true zero, so all arithmetic, including ratios, is licensed. Most StyleCraft quantities live here: `quantity`, `unit_price`, `line_revenue`, `spend`, `impressions`, `clicks`. A campaign with $40,000 of spend really did spend twice as much as one with $20,000.

Table 3.2 summarizes the four levels with StyleCraft examples. The classification is a working tool rather than a law of nature — Velleman and Wilkinson (1993) document its limits — but as a first pass over an unfamiliar dataset, "what level is this column?" remains among the most productive questions an analyst can ask.

Table 3.2

Levels of measurement with StyleCraft examples

Level	What it carries	StyleCraft examples	Permissible summaries	Not permissible
Nominal	Category labels, no order	acquisition_channel, channel, gender, store_name, customer_id	Counts, proportions, mode	Any arithmetic; averaging identifiers
Ordinal	Ordered categories, undefined distances	loyalty_tier, age_band	Median, percentiles, order comparisons ("VIP or above")	Means and differences (strictly)
Interval	Meaningful distances, no true zero	order_date, signup_date	Differences (durations), medians, means of differences	Ratios ("date A is twice date B")
Ratio	Meaningful distances and a true zero	quantity, unit_price, line_revenue, spend, clicks	All arithmetic, including ratios and rates	—

3.4.1 From Measurement Levels to Stored Data Types

Levels of measurement describe what a variable means. Data types describe how software stores it, and the two classifications do not coincide. In pandas, the Python library used throughout this guide, a column arrives as an integer (`int`), a decimal (`float`), a Boolean (`bool`), a `datetime`, or — for anything textual — a general-purpose type whose name depends on the version in use. Versions before pandas 3.0 report text columns as the generic `object` type; pandas 3.0 changed the default so that text is inferred as a dedicated `str` type (pandas development team, n.d.). The interpretive situation is identical under either name. Nothing in the stored type records whether a text column is nominal or ordinal, whether an integer is a quantity or an identifier, or whether a float is a price or a percentage. The meaning lives in the analyst's head, or in the data dictionary, but not in the file.

The mismatch runs in both directions and both directions cause errors. A meaningful order can be invisible to the software: `loyalty_tier` arrives as plain text, so sorting it yields alphabetical order (Backstage, Insider, None, VIP) rather than tier order, and "VIP or above" cannot be expressed until the analyst declares the ordering, which pandas supports through ordered categorical types (the lab in Section 3.10 does exactly this). Conversely, an

order the software offers can be meaningless: a numeric-looking identifier will happily produce an average if asked. Furthermore, Booleans quietly straddle the classification: `email_opt_in` is a nominal yes/no in meaning, yet because True is stored as 1 and False as 0, its mean is a legitimate and useful quantity — the opt-in rate. The habit this subsection recommends is a small, repeatable check performed on every new dataset: list the stored types, state the intended measurement level of each column, and reconcile the two before computing anything. That check is the heart of Lab 3.1.

CONCEPT

Why the Classification Matters Later

The level of measurement governs three choices that recur in every subsequent chapter: which summary statistics are meaningful (Chapter 5), how a variable must be encoded before modeling — for example, nominal variables becoming dummy variables in regression (Chapter 7) — and which chart types are appropriate, since bar charts suit categories while histograms and scatterplots suit quantities (Chapter 12). Misclassifying a variable at this stage propagates into wrong summaries, wrong encodings, and wrong charts later.

Source: Course concept developed for this guide, informed by Stevens (1946) and Velleman and Wilkinson (1993).

3.5 Metrics, KPIs, Dimensions, and Features

The previous section classified variables by what they carry. This section classifies analytic work by the role a quantity or an attribute plays in it, because one business concept — and sometimes one underlying field — can appear in four different positions. The four role words — metric, KPI, dimension, and feature — are used constantly in marketing organizations, frequently interchangeably, and the confusion is not harmless: calling every metric a "KPI" inflates dashboards until nothing is key, and confusing dimensions with features muddles the transition from reporting to modeling in Part II of this guide.

A metric is a quantity computed from data by an explicit rule, at a stated grain, for the purpose of description or comparison. Average order value, repeat purchase rate, conversion rate, return on ad spend, and cost per acquisition are metrics. Standard references catalog hundreds of marketing metrics precisely because the discipline's quantities are conventional, not natural: each exists only as its definition (Farris et al., 2010). Appendix B collects the definitions used in this guide.

A key performance indicator is a metric that an organization has elevated: attached to a target, assigned an owner, reviewed on a cadence, and connected to strategy. The KPI literature insists that the K is doing real work — an organization can compute hundreds of metrics but can attend to only a handful of KPIs, and indicators that merely describe results differ from those that anyone can act on (Parmenter, 2015). Elevation also has a cost that Chapter 2 examined: once a metric becomes a target, behavior reorganizes around it and the measure degrades as evidence (Campbell, 1979; Strathern, 1997). Choosing KPIs is therefore a managerial act with side effects, not a technical selection.

A dimension is an attribute used to slice, group, or filter metrics: `store_type`, `acquisition_channel`, `age_band`, `category`, month. Dimensions are usually nominal or ordinal variables drawn from the dimension tables of Section 3.3, and "metric by dimension" — repeat rate by store type, revenue by channel — is the basic grammatical unit of descriptive analytics and of every dashboard in Part III.

A feature is a variable prepared as an input to a model: recency in days, order frequency, discount share of purchases, tenure, opt-in flags. The word belongs to Part II of this guide, where models predict spend (Chapter 8) or churn (Chapter 9) from features. Features are frequently engineered — computed from raw columns at the model's unit of analysis — and feature representation and engineering are often among the most consequential parts of

model development (Domingos, 2012). A dimension used for slicing today often reappears, encoded, as a feature tomorrow.

One asymmetry among the four is worth naming, because a compact table can obscure it. Metrics and KPIs are ordinarily computed quantities: they do not sit in the data waiting to be read, they are produced by a rule. Dimensions and features are roles played by fields that already exist, or that have been engineered into existence. So these are not four uses of one column. They are four positions that one business concept can occupy in analytic work, and only some of those positions are occupied by a column. Table 3.3 arranges the four around a single StyleCraft concept, repeat purchase, to make the distinctions concrete.

Table 3.3

One concept, four roles: repeat purchase at StyleCraft

Role	What it is	Repeat-purchase example	Where it appears in this guide
Metric	A quantity computed by an explicit rule at a stated grain	Repeat purchase rate: repeat customers ÷ active customers, trailing 12 months	Chapter 2 lab; Chapter 5
KPI	A metric elevated with a target, owner, and review cadence	Repeat rate ≥ 45%, owned by the CRM manager, reviewed monthly against the expansion decision	Chapter 2 (Section 2.12); Chapter 13 dashboards
Dimension	An attribute used to slice or filter metrics	Repeat rate by store_type, by acquisition_channel, by age_band	Chapter 5 cross-tabs; Chapters 12–13
Feature	A variable prepared as a model input	repeat_flag or purchase frequency as an input to the churn model	Chapters 6–9

3.5.1 The Anatomy of a Ratio Metric

Most marketing metrics are ratios, and the Chapter 1 and Chapter 2 labs have already shown that ratios conceal their most consequential choices in the denominator (Exercise 1.10, item 2, "The Averaged Average," and Section 2.9.4). This subsection generalizes the lesson: a ratio metric is not fully defined until four elements are written down.

First, the numerator: what is being counted or summed, under what rule. Second, the denominator: the population, exposure, or base against which the numerator is expressed. For many customer- or event-based rates the denominator also reveals the entity being counted, but that is a convenience rather than a rule — return on ad spend divides by dollars of spend while the unit of analysis is usually the campaign — so the metric's unit of analysis should still be stated separately. Third, the time window: over what period both are accumulated, and as of when. Fourth, the filters: which records are included or excluded — channels, statuses, test customers, returns, cancelled orders.

The Section 3.1 dispute is precisely a disagreement across these four elements. All three teams counted "customers with two or more purchases" in the numerator; they diverged on the denominator (customers with at least one completed order in the available company history, versus customers active in the window, versus customers who shopped in a store), on the window (lifetime versus trailing twelve months), and on the filters (all channels versus stores only). Written this way, the dispute stops being mysterious and becomes tractable: the organization does not need to decide which number is true; it needs to decide which definition serves the decision at hand, and to record the definition so the question stays settled. Section 3.8 shows where such definitions live, and the lab in Section 3.10 computes all three from the same miniature dataset.

DEFINITION

Metric

A metric is a quantity computed from data by an explicit rule for description or comparison. A ratio metric is completely specified only when its numerator, denominator, time window, and filters are stated.

Source: Adapted from Farris et al. (2010).

DEFINITION

Key Performance Indicator (KPI)

A key performance indicator is a metric that an organization has elevated to guide action: connected to strategy, attached to a target, assigned an owner, and reviewed on a regular cadence. Because targets reshape behavior, the selection of KPIs is a managerial decision with side effects, not a purely technical one.

Source: Adapted from Parmenter (2015), Campbell (1979), and Strathern (1997).

DEFINITION

Dimension and Feature

A dimension is an attribute used to slice, group, or filter metrics in descriptive analysis and dashboards. A feature is a variable prepared as an input to a predictive model, often engineered from raw columns at the model's unit of analysis. The same underlying field can serve as a dimension in reporting and, suitably encoded, as a feature in modeling.

Source: Adapted from Kimball and Ross (2013), Provost and Fawcett (2013), and Domingos (2012).

3.6 Measurement Validity and Reliability

So far this chapter has treated measurement decisions as choices among defensible alternatives. This section examines how measurement fails. The classical vocabulary distinguishes two failure modes: a measure can fail to capture the concept it claims to capture, which is a validity problem, or it can capture something inconsistently, which is a reliability problem (Carmines & Zeller, 1979).

Validity asks: does this measure measure what we say it measures? The question sounds abstract until it is aimed at a familiar column. Does `email_opt_in` measure interest in the brand, or does it measure the aggressiveness of the signup flow in force when the customer joined? Does `loyalty_tier` measure loyalty, or does it measure enrollment — a customer's willingness to complete a form — which the Suburban Occasion shopper profile suggests can diverge sharply from actual attachment to the brand? Does a click measure interest, or curiosity, or a misplaced thumb? In each case the column is a valid measure of something; the validity question is whether it is a valid measure of the concept the analysis names. Marketing researchers develop the question into finer distinctions — content validity (does the measure cover the concept?), construct validity (does it behave as the concept should?), and criterion validity (does it predict what the concept should predict?) — and while formal validation belongs to research methods courses, the working analyst needs at least the reflex of asking the question (Churchill, 1979; Carmines & Zeller, 1979).

Reliability asks: would the measure give the same answer under repetition? A survey question that respondents interpret differently on different days is unreliable; so is a "store visits" figure produced by a sensor that undercounts on crowded days, or a revenue feed that a source system restates for forty-eight hours after each day closes.

Reliability is a precondition for validity — a measure that will not sit still cannot be measuring the concept well — but it is not a substitute for it: a scale that is consistently wrong by two kilograms is perfectly reliable and invalid.

3.6.1 Proxy Measures

Marketing analytics runs on proxies: measures adopted to stand in for concepts that are not directly observable. Repeat purchase rate stands in for loyalty; clicks stand in for interest; email opens stand in for attention; `loyalty_tier` stands in for relationship depth. Proxies are not a failure of practice — they are unavoidable, since loyalty itself is never a column — but each proxy imports assumptions, and the assumptions can fail silently. A striking demonstration comes from outside marketing: a widely used healthcare algorithm used medical costs as a proxy for medical need, and because less money was historically spent on Black patients at the same level of illness, the proxy systematically understated their need (Obermeyer et al., 2019). The proxy was convenient, computable, and consequentially wrong. Marketing's proxies deserve the same scrutiny: a repeat-rate proxy for loyalty, for example, will understate the loyalty of customers whose category purchase cycle is naturally long — the occasionwear buyer may be devoted and still shop twice a year. Section 3.13 returns to the ethics of this point.

Finally, validity connects back to Chapter 2's discussion of metric gaming (Section 2.12). When a proxy becomes a target, people optimize the proxy rather than the concept, and the gap between them widens: enrolling every walk-in customer in the loyalty program raises `loyalty_tier` statistics while measuring loyalty less validly than before. Validity is therefore not a one-time property of a measure; it can be eroded by the very act of managing to it. That is the risk StyleCraft is about to take on, since Section 3.1 established that whichever repeat-rate definition wins will become a team target.

DEFINITION

Validity

Validity is the degree to which a measure captures the concept it is intended to capture. In marketing analytics the question is most often aimed at proxies: whether repeat purchase validly measures loyalty, whether clicks validly measure interest, whether tier enrollment validly measures relationship depth.

Source: Adapted from Carmines and Zeller (1979) and Churchill (1979).

DEFINITION

Reliability

Reliability is the degree to which a measure produces consistent results under repeated or equivalent measurement. Reliability is necessary but not sufficient for validity: a consistently biased measure is reliable and invalid.

Source: Adapted from Carmines and Zeller (1979).

3.7 Data Quality: Dimensions and Common Defects

Validity and reliability concern whether the right thing is being measured. Data quality concerns whether the recorded values are fit for use at all. The two concerns are related but distinct: a dataset can record a perfectly valid measure full of typing errors, and a pristine dataset can record an invalid proxy flawlessly. Research on data quality converges on the view that quality is multidimensional and defined by fitness for the consumer's use, not by any single property of the file (Wang & Strong, 1996). The six dimensions used below are an operational synthesis for this guide, informed by Wang and Strong's consumer-centered framework and by Redman's (1998) account of how quality failures propagate through an enterprise; neither source states this exact six-item taxonomy.

Accuracy asks whether values reflect reality: a price recorded as $2,400 for a $24.00 top is inaccurate, most likely through a decimal error at entry. Completeness asks whether values that should be present are present — and, more subtly, what an absence means. In StyleCraft's `transactions`, a missing `campaign_id` may be legitimately null (the order was not attributable to any campaign) or truly missing (the attribution failed); the two absences look identical in the file and mean opposite things for any campaign analysis. Consistency asks whether the same fact is recorded the same way everywhere: a channel column containing "Online," "online," and "web" is inconsistent, and a naive group-by will report three channels where one exists. Uniqueness asks whether each real-world event appears exactly once: fully duplicated order lines inflate revenue, and near-duplicates — the same order re-exported with a different timestamp — inflate it while evading the obvious duplicate check. Timeliness asks whether the data is current enough for the decision, and whether late-arriving records will restate figures already reported. Finally, conformity (sometimes called validity in the data-quality literature, a regrettable collision with Section 3.6's usage) asks whether values respect their declared type, format, and allowed range: an `order_date` before the customer's `signup_date`, a quantity of 999, a price stored as the string "$45.00" rather than the number 45.00.

Table 3.4 summarizes the six dimensions with StyleCraft examples. Readers may notice that the examples are suspiciously specific. They are: the StyleCraft raw files used in Chapter 4 ship with a documented catalog of planted defects spanning exactly these categories, and Chapter 4's lab is the work of finding and remedying them. This chapter's job is to supply the vocabulary; the next chapter's job is the repair.

Table 3.4

Data quality dimensions with StyleCraft examples

Dimension	Question it asks	StyleCraft example (Chapter 4 preview)
Accuracy	Do values reflect reality?	Prices outside the $24–$90 catalog band; implausible quantities such as 999
Completeness	Are expected values present, and what does an absence mean?	Missing age_band; campaign_id that is legitimately null versus truly missing
Consistency	Is the same fact recorded the same way everywhere?	"Online" / "online" / "web" in the channel column; "Occasionwear" / "occasion wear"
Uniqueness	Does each event appear exactly once?	Fully duplicated order lines; near-duplicates re-exported with new timestamps
Timeliness	Is the data current enough, and stable once reported?	Recent stores with short histories; feeds that restate recent days
Conformity	Do values respect type, format, and allowed ranges?	order_date before signup_date; numbers arriving as strings with currency symbols

Two habits follow for the working analyst. First, data quality is assessed against use: a duplicate rate that is immaterial for a company-level revenue trend may be fatal for a customer-level churn model, so "is this data good?" is always shorthand for "is this data good enough for this decision?" (Wang & Strong, 1996; Redman, 1998). Second, known defects belong in the documentation, not in the analyst's memory. A column's data dictionary entry, introduced next, has a field for exactly this purpose.

DEFINITION

Data Quality

Data quality is the degree to which data is fit for its intended uses, assessed along dimensions including accuracy, completeness, consistency, uniqueness, timeliness, and conformity to declared formats and ranges. Quality is judged relative to the decision the data supports, not as an absolute property of a file.

Source: Course synthesis for this guide, informed by Wang and Strong (1996) and Redman (1998).

3.8 The Data Dictionary

The preceding sections have accumulated a list of things that must be written down: the grain of each table, the intended measurement level of each column, the operational definition behind each metric, the known quality issues. This section introduces the document where they are written: the data dictionary. A data dictionary is a shared, maintained reference that records, for every table and column in a data family — and, in the extended form this guide recommends, for every named metric — what it means, how it is typed, what values it may take, where it comes from, and what is known to be wrong with it (Kimball & Ross, 2013).

The case for the dictionary is the case against private knowledge. Without a dictionary, every column's meaning lives in the heads of whoever loaded it last, definitions drift as staff turn over, and disputes like Section 3.1's are not resolved but merely adjourned. With a dictionary, a new analyst can be productive in days rather than months, AI assistants can be given accurate context instead of guessing at column meanings (Section 3.9), and a metric definition, once agreed, stays agreed. The dictionary is to a data family what the analytic specification (Section 2.5) is to a single project: a small written discipline that prevents large silent failures.

A useful column entry records at least eight things: the column name; the table and grain it belongs to; a one-sentence plain-language definition; the stored data type; the intended level of measurement; the allowed values or range; the source system or derivation rule; and known issues. Table 3.5 shows entries for five StyleCraft columns in this format. The entries are deliberately short — a dictionary that is painful to read will not be read — and deliberately candid: the known-issues field for `campaign_id` records the null-versus-missing ambiguity rather than hiding it.

Table 3.5

Data dictionary entries for selected StyleCraft columns

Column (table)	Definition	Type / Level	Allowed values	Source / Known issues
loyalty_tier (customers)	Customer's current tier in the VIP loyalty program	categorical / ordinal	None < Insider < VIP < Backstage	Loyalty platform; measures enrollment, not attachment — see Section 3.6.1
age_band (customers)	Self-reported age at signup, bucketed	categorical / ordinal	18–24, 25–34, 35–44, 45–54, 55+	Signup form; optional, so some values missing; banded for privacy
discount_pct (transactions)	Discount applied to the line, as a proportion	float / ratio	0 to 0.80	E-commerce and POS; 0 means full price; small share missing in raw file
campaign_id (transactions)	Campaign to which the line is attributed	str / nominal (identifier)	Valid campaign key or null	Attribution pipeline; null is legitimate (unattributed) but some nulls are attribution failures
line_revenue (transactions)	quantity × unit_price × (1 − discount_pct)	float / ratio	≥ 0	Derived; recompute to verify rather than trusting the stored value

3.8.1 Metrics Belong in the Dictionary Too

Column entries are necessary but not sufficient, because the disputes that damage organizations are usually about computed quantities rather than raw columns. The extended dictionary therefore includes metric entries, each recording the four elements of Section 3.5.1 — numerator, denominator, window, filters — together with the grain, the owner, and the decision the metric informs. The last two fields do work the first four cannot. An unowned definition has no one to defend it when a new dashboard quietly redefines it, and a definition with no stated decision use cannot be evaluated, because fitness for use is the only standard a metric can be judged against (Section 3.7).

Table 3.6 writes the three repeat purchase rates from the opening case in exactly this format. Set side by side, the three entries make the meeting's confusion almost impossible to reproduce: the numbers differ because the definitions differ, and the definitions now differ in writing.

Table 3.6

Three metric definitions behind one metric name

Element	"Lifetime" version (Retail)	"Trailing 12M" version (CRM)	"Store" version (Finance)
Numerator	Customers with ≥ 2 orders ever	Customers with ≥ 2 orders in window	Customers with ≥ 2 store orders in window
Denominator	All customers with ≥ 1 completed order in the available history	Customers with ≥ 1 order in window	Customers with ≥ 1 store order in window
Window	Entire available history to the analysis date	Trailing 12 months to the analysis date	Trailing 12 months to the analysis date
Filters	All channels	All channels	Store channel only
Grain	Customer	Customer	Customer
Metric owner	Vice president of retail	CRM manager	Finance, retail economics
Decision or business use	Long-run brand narrative; not suitable for period-over-period comparison	Headline retention tracking; baseline for the expansion review	Economics of the physical footprint; store-level go/no-go
Value in the Lab 3.1 preview	0.60	0.43	0.20

One more distinction is worth drawing before the machinery is put to use. The dictionary does not decide which definition is right; it makes the alternatives comparable so that the decision-maker can choose. Section 3.11 argues that for the expansion decision, the trailing-window definition is the most defensible headline, with the store version reported alongside it — but that is a judgment about the decision, reached with the dictionary's help, not a fact the dictionary contains.

DEFINITION

Data Dictionary

A data dictionary is a shared, maintained document that records, for each table, its grain; for each column, its definition, type, level of measurement, allowed values, source, and known issues; and, in the extended form used in this guide, for each named metric, its numerator, denominator, window, filters, grain, owner, and the decision it informs.

Source: Adapted from Kimball and Ross (2013); the metric extension is an operational course convention.

3.9 AI as a Measurement and Documentation Assistant

The previous sections described work — typing columns, drafting definitions, assembling dictionaries — that is exactly the kind of structured, patterned labor AI assistants accelerate well. Consistent with the guide's stance (Sections 1.9 and 2.8), this section identifies where the assistance genuinely helps and where the analyst must not delegate.

The genuine help is substantial. Given a sample of rows, an assistant can draft a plausible data dictionary in minutes: inferring that a column is categorical, guessing allowed values from the observed ones, and writing serviceable one-sentence definitions. It can propose the four-element definition of a vague metric ("what would

'active customer' need to mean for this table?"), flag columns whose names and contents disagree, and translate a schema into plain language for a stakeholder. For the tedious first draft of documentation, the assistant is close to ideal.

The risks are equally specific. First, inference from a sample is not knowledge of a system: an assistant seeing only values 0 to 0.4 in `discount_pct` will confidently document the allowed range as 0 to 0.4, when the true business rule permits up to 0.8. Second, assistants complete ambiguity fluently: asked to define repeat purchase rate, an assistant will produce a definition — some definition — and a definition that arrives fluently is easily mistaken for the organization's definition, which is precisely how the Section 3.1 problem gets automated rather than solved. Third, assistants inherit the file's silences: nothing in the data reveals that `campaign_id` nulls come in two kinds, so no assistant will document it unprompted. The general pattern is the one this guide keeps encountering: the assistant drafts, and the analyst remains responsible for everything the data does not itself contain — business rules, definitional authority, and known history.

The countermeasure is the predict-then-verify habit applied to schema work. Before asking an assistant to infer types, write down the expected type and level of each column; then compare. Before accepting a drafted metric definition, check each of the four elements against the decision the metric serves. Moreover, dictionary drafts should be audited with the source-system owners, who hold the knowledge no file contains.

AI IN PRACTICE

Draft the Dictionary, Audit the Draft

A productive pattern: paste a small sample of rows (never confidential data; the synthetic StyleCraft files are safe) and prompt, "Draft a data dictionary entry for each column: definition, type, level of measurement, allowed values, and one plausible known issue. Mark any column where you are guessing."

Then audit the draft: verify allowed values against business rules rather than observed values, correct the measurement levels, and add the issues only humans know. Document the exchange per Appendix D. The assistant's "known issue" guesses are often generic, but occasionally an assistant flags a real ambiguity the analyst had stopped seeing.

3.10 Hands-On Application in Python and Google Colab

The preceding sections were conceptual. This section moves from concepts to practice by putting the chapter's two central skills — reconciling stored types with measurement levels, and defining metrics precisely — to work on a miniature version of the opening case. As in Chapters 1 and 2, the lab uses a compact dataset created directly in Python so that every number can be verified by hand; it is constructed to reproduce, in miniature, the pattern of the three numbers from the Section 3.1 meeting. Later chapters load StyleCraft's full data family from the companion repository, accompanied by the student-facing data dictionary this chapter has been building toward.

3.10.1 Lab 3.1: Reading the Types Before Trusting the Table

The lab begins with a ten-customer preview of StyleCraft's `customers` table and an eighteen-order preview of its orders. Before running anything, apply predict-then-verify to the types themselves. The predictions belong on paper, not in memory.

VERIFICATION CHECK

Before you run: `customer_id` should be text; `signup_date` should be a date; `acquisition_channel` should be an unordered category; `age_band` and `loyalty_tier` should be ordered categories; `email_opt_in` should be Boolean. The customers frame should report ten rows and six columns, the orders frame eighteen rows and three columns. Every repeat purchase rate, being a proportion, must fall between 0 and 1.

After you run: pandas should report a Boolean type for `email_opt_in` and a single generic text type for all five remaining columns — `object` in versions before 3.0, `str` in 3.0 and later. Five of the six predictions about *meaning* will therefore appear to fail, which is the point of the exercise rather than an error. After the conversions in Code 3.2, the highest tier present should be VIP and four customers should be Insider or above. The three rates in Code 3.4 should be 0.60, 0.43, and 0.20.

Investigate if: any rate falls outside 0 and 1; either frame reports a shape other than (10, 6) or (18, 3); or a rate changes when the notebook is rerun on a later date, which would indicate a window boundary that depends on the day of execution rather than on the stated analysis date.

Code 3.1. Create the customer preview

```
import pandas as pd

print("pandas version:", pd.__version__)

customers = pd.DataFrame({
    "customer_id": ["C01", "C02", "C03", "C04", "C05",
                    "C06", "C07", "C08", "C09", "C10"],
    "signup_date": ["2024-05-12", "2024-10-01", "2025-09-20",
                    "2025-07-30", "2024-08-15", "2025-12-28",
                    "2025-10-05", "2024-07-01", "2026-05-22",
                    "2025-01-10"],
    "acquisition_channel": ["Instagram", "TikTok", "In-Store",
                            "Instagram", "Paid Search", "Referral",
                            "Email", "In-Store", "TikTok",
                            "Influencer"],
    "age_band": ["18-24", "25-34", "45-54", "25-34", "35-44",
                 "18-24", "25-34", "55+", "18-24", "35-44"],
    "loyalty_tier": ["Insider", "None", "None", "VIP", "None",
                     "Insider", "None", "None", "None", "Insider"],
    "email_opt_in": [True, True, False, True, False,
                     True, True, False, True, True]
})

print(customers.shape)
customers.dtypes
```

Expected output: the shape (10, 6); `email_opt_in` reports `bool`; `customer_id`, `signup_date`, `acquisition_channel`, `age_band`, and `loyalty_tier` all report the same generic text type — `object` under pandas 2.x, `str` under pandas 3.0 and later.

The version line is printed deliberately. Colab environments change underneath a course, and the name of the text type is one of the things that changed: pandas 3.0 made a dedicated `str` type the default, where earlier versions reported `object` (pandas development team, n.d.). The interpretive question is identical under either name, which is the lesson.

Compare the output with the predictions. The Boolean survived. Everything else arrived as one undifferentiated text type, including `signup_date`, which is currently a string that merely looks like a date. Nothing in the file records that `loyalty_tier` is ordered, that `age_band` is banded, or that `customer_id` is a label rather than data. The stored types have discarded most of the measurement information, exactly as Section 3.4.1 warned.

3.10.2 Declaring the Measurement Levels

The next cell restores the discarded meaning. Dates become datetimes; `acquisition_channel` becomes an unordered category; `age_band` and `loyalty_tier` become ordered categories, so that the software finally knows what the loyalty program knows. All four conversions are performed here, because a reconciliation that stops halfway leaves exactly the columns most likely to be misinterpreted still undeclared.

Code 3.2. Align stored types with intended measurement levels

```
# Interval: a real date, not a string that looks like one
customers["signup_date"] = pd.to_datetime(customers["signup_date"])

# Nominal: an unordered category
customers["acquisition_channel"] = pd.Categorical(
    customers["acquisition_channel"], ordered=False
)

# Ordinal: ordered categories, declared in the order the business uses
band_order = ["18-24", "25-34", "35-44", "45-54", "55+"]
customers["age_band"] = pd.Categorical(
    customers["age_band"], categories=band_order, ordered=True
)

tier_order = ["None", "Insider", "VIP", "Backstage"]
customers["loyalty_tier"] = pd.Categorical(
    customers["loyalty_tier"], categories=tier_order, ordered=True
)

print(customers.dtypes)

# Ordinal operations are now meaningful
print(customers["loyalty_tier"].max())
print((customers["loyalty_tier"] >= "Insider").sum())

# A permissible mean: the Boolean straddle from Section 3.4.1
print(customers["email_opt_in"].mean())
```

Expected output: `signup_date` reports a datetime type; the three converted columns report `category`; the highest tier present is VIP; four customers are Insider or above; the opt-in rate is 0.7.

Three verification checks close the loop, and each one demonstrates a different way the declaration earns its keep. First, sorting by `loyalty_tier` now yields tier order rather than alphabetical order; before the conversion it would have placed “None” between “Insider” and “VIP,” a quietly wrong ordering that a chart would have faithfully displayed. Second, `customers["age_band"].mean()` raises a `TypeError`, because averaging an ordered category is not licensed — and note that the declaration, not luck, is what produces the error. Had ages been stored as numeric midpoints, the impermissible average would have computed without complaint. Third, `customers["acquisition_channel"].max()` also raises, with the message that the category is not ordered. Declaring a variable nominal is not bookkeeping; it installs a guardrail that Section 3.4 could only describe.

3.10.3 One Metric Name, Three Numbers

The lab now reconstructs the opening dispute. A compact orders table records eighteen orders by the ten customers over StyleCraft's twenty-four-month window. The analysis date is June 30, 2026, so the trailing-twelve-month window opens on July 1, 2025.

Code 3.3. Create the order preview

```
orders = pd.DataFrame({
    "customer_id": ["C01", "C01", "C01", "C02", "C02", "C03",
                    "C04", "C04", "C04", "C05", "C06", "C06",
                    "C07", "C08", "C08", "C09", "C10", "C10"],
    "order_date": pd.to_datetime([
        "2024-08-10", "2025-09-14", "2026-02-02", "2024-11-05",
        "2025-01-20", "2025-10-03", "2025-08-19", "2025-12-24",
        "2026-05-30", "2024-09-01", "2026-01-15", "2026-01-28",
        "2025-11-11", "2024-07-22", "2026-04-10", "2026-06-06",
        "2025-02-14", "2025-05-02"]),
    "channel": ["Store", "Store", "Online", "Online", "Online",
                "Store", "Online", "Store", "App", "Store",
                "Store", "Store", "Online", "Store", "Store",
                "App", "Store", "Store"]
})

orders.shape
```

Expected output: (18, 3).

Before computing, predict — and be careful about what kind of claim the prediction is. After inspecting these eighteen rows, we expect the lifetime version to be the largest, because every customer who ever repurchased keeps counting, and the store-only version to be the smallest, because it discards two channels from the numerator and the denominator alike. That ordering is a prediction about these particular records, not a mathematical property of the three definitions. A lifetime denominator grows along with its numerator, and filtering to stores can raise a rate as easily as lower it, depending on which customers survive the filter. Predicting the ordering and then noticing that it was contingent is itself the predict-then-verify lesson.

Code 3.4. Compute the three repeat purchase rates

```
analysis_date = pd.Timestamp("2026-06-30")
window_start = pd.Timestamp("2025-07-01")

# Definition A (Retail): all history up to the analysis date,
# customers with at least one order on record
history_to_date = orders[orders["order_date"] <= analysis_date]
lifetime_orders = history_to_date.groupby("customer_id").size()
rate_a = (lifetime_orders >= 2).mean()

# Definition B (CRM): trailing 12 months, customers active in the window
in_window = orders[
    orders["order_date"].between(window_start, analysis_date)
]
window_orders = in_window.groupby("customer_id").size()
rate_b = (window_orders >= 2).mean()

# Definition C (Finance): store channel only, trailing 12 months
store_window = in_window[in_window["channel"] == "Store"]
store_orders = store_window.groupby("customer_id").size()
rate_c = (store_orders >= 2).mean()

print(round(rate_a, 2), round(rate_b, 2), round(rate_c, 2))
```

Expected output: 0.60 0.43 0.20.

Two details of this cell are places where a plausible implementation drifts away from the written definition. The first is Definition A's denominator. The lifetime rate is computed by grouping the orders table, so its base is customers with at least one order on record — which is why Table 3.6 states the denominator that way rather than as "all customers on file." In this preview every customer has ordered, so both phrasings give 0.60; in a real CRM they would not, because customers who signed up and never purchased sit in `customers` and never appear in `orders`. A repeat rate among purchasers is a defensible metric, but it has to be the stated one. Reindexing the counts against the full `customers` table, so non-purchasers enter as zeros, is a different metric and belongs in the dictionary under a different name.

The second is the window. Definitions B and C are bounded on both sides, using `between` with an explicit `analysis_date` rather than keeping everything at or after the window start. A one-sided filter admits future-dated and mistyped orders, and it quietly makes the reported rate depend on the day the notebook happens to run. A trailing window needs a beginning and an "as of" date, which is why the third element of a metric definition is a window rather than a start date. Note that Definition A takes the same "as of" bound even though it has no start: a lifetime metric is still measured up to a date, and imposing that bound is what makes the code enforce the definition the dictionary publishes. These eighteen rows contain no future-dated orders, so the figure stays 0.60 — which is precisely the situation in which a missing boundary goes unnoticed until the day it does not.

The three slides from the meeting have now been reproduced from a single, fully visible dataset of eighteen rows. It is worth pausing on what the code makes explicit that the meeting left implicit: each definition filters a different set of rows and groups over a different base of customers. Definition A's denominator is all ten customers; Definition B's is the seven customers active in the window; Definition C's is the five customers with a store order in the window. The numerators shrink in parallel: six lifetime repeaters, three in-window repeaters, one in-window store repeater — C06, whose two January store visits make the point that a base of five customers turns a single person into twenty percentage points, a small-denominator caution that Chapter 5 develops.

3.10.4 Writing the Dictionary Entry

The final step converts the computation into documentation, building the metric portion of a data dictionary as a small table of its own. The frame carries the owner and the decision use alongside the four defining elements, in the same wording as Table 3.6: a definition without an owner has no one to defend it, and a definition without a stated use cannot be judged fit for anything. Note also that the recorded values are the computed variables rather than numbers typed in by hand. Hardcoding 0.60, 0.43, and 0.20 would work today and would silently stop matching the analysis the first time a definition changed — the documentation drifting from the computation is the same failure this chapter opened with, in miniature.

Code 3.5. Record the three definitions as dictionary entries

```
metric_dictionary = pd.DataFrame({
    "metric_name": ["repeat_rate_lifetime", "repeat_rate_t12m",
                    "repeat_rate_store_t12m"],
    "numerator": ["customers with >= 2 orders ever",
                  "customers with >= 2 orders in window",
                  "customers with >= 2 store orders in window"],
    "denominator": ["all customers with >= 1 completed order "
                    "in the available history",
                    "customers with >= 1 order in window",
                    "customers with >= 1 store order in window"],
    "window": ["entire available history to the analysis date",
               "trailing 12 months to the analysis date",
               "trailing 12 months to the analysis date"],
    "filters": ["all channels", "all channels", "store channel only"],
    "grain": ["customer", "customer", "customer"],
    "owner": ["Vice president of retail", "CRM manager",
              "Finance, retail economics"],
```

```
    "decision_use": ["long-run brand narrative; not suitable for "
                     "period-over-period comparison",
                     "headline retention tracking; baseline for the "
                     "expansion review",
                     "economics of the physical footprint; store-level "
                     "go/no-go"],
    "value_in_preview": [round(rate_a, 2), round(rate_b, 2),
                         round(rate_c, 2)]
})

metric_dictionary
```

Expected output: a table with three rows, one per definition, matching Table 3.6.

The dictionary does not merely record the past computation; it changes future behavior. Any analyst — or any AI assistant supplied with this table as context — asked for "the repeat purchase rate" must now confront the fact that three exist, and must choose one by name. Ambiguity has been converted into a menu.

3.10.5 One More Verification: Hand-Check the Middle Number

The final planned check is a hand count of Definition B, the number most likely to become the headline. Customers with at least one order between July 1, 2025 and June 30, 2026: C01, C03, C04, C06, C07, C08, and C09 — seven customers. Of these, customers with at least two orders in the window: C01 (September 2025 and February 2026), C04 (August, December, May), and C06 (two January orders) — three customers. Three divided by seven is approximately 0.43, matching the computed value. As in Chapter 2, a single hand-checked figure does not prove the code correct, but it catches reversed filters, wrong comparison operators, and off-by-one window boundaries — and with ten customers it costs one minute, which is precisely why this guide's labs stay small enough to count.

3.11 Marketing Interpretation and Managerial Insight

The previous section produced three verified numbers and a dictionary that distinguishes them. This section asks what StyleCraft should do with them before the October 15 review, which is where measurement work either earns its keep or remains a technical exercise.

A weak response to the meeting would be to average the three figures, or to let the loudest stakeholder's number stand, or to declare the discrepancy a "data issue" and commission a systems project. Each move mistakes a definitional question for an arithmetic or technical one. The numbers do not disagree; the definitions do, and definitions are chosen, not fixed.

A stronger response reasons from the decision, in the manner of Chapter 2. The live decision is the store expansion, and the decision-relevant question is whether recently acquired customers are forming a repurchase habit now. The lifetime definition is nearly useless for this: it is dominated by the accumulated history of the oldest, urban-acquired customers, it can only rise slowly or fall slowly, and it will register a change in current behavior years late. It is not wrong; it is unresponsive. The trailing-twelve-month definition is built for the question — it holds the window fixed, so movement in the number reflects movement in behavior — and it is the natural headline metric, with one caveat the dictionary should record: customers acquired late in the window have had little time to repurchase, a mechanical depressant that the same-age cohort comparison promised in Chapter 2 is designed to remove. The store-only definition, finally, answers finance's narrower and legitimate question about the economics of the physical footprint, and should be reported alongside the headline rather than suppressed — labeled, so that a 20 and a 43 can no longer share a slide unexplained.

The managerial recommendation that follows is modest and concrete: adopt `repeat_rate_t12m` as StyleCraft's headline definition, assign the CRM manager as its owner, report `repeat_rate_store_t12m` beside it in expansion reviews, retire the lifetime figure from decision documents, and publish all three entries in the data dictionary with their owners and decision uses attached. One further recommendation belongs with them, because Section 3.1 established that the chosen definition will also become a CRM performance target. The dictionary entry should carry the gaming risk explicitly — enrollment tactics that manufacture second purchases will

raise the measured rate without creating loyalty — and the review should pair the headline rate with at least one metric that such tactics would not move.

Notice what the recommendation does not claim. It does not claim the headline number is good or bad, and it does not resolve the expansion question. Measurement work rarely answers the business question by itself. What it does is subtler and, over time, more valuable: it ensures that when the organization argues, it argues about the business rather than about what its own words mean.

3.12 Business Analytics in Practice

The previous section resolved a definitional dispute inside a fictional retailer. This section turns from the fictional case to real organizational practice, and to what the same dispute costs the organizations that live it: where metric definitions are regulated, where they are contested between departments, and where a single parameter in an advertising platform quietly reallocates a budget. The three situations differ enormously in scale. The mechanism is the one this chapter has been describing throughout.

3.12.1 "Active Users," in Public

Consumer technology companies are valued substantially on user counts, which means that a phrase like *monthly active users* is not internal vocabulary but a reported figure investors act on. It is also, like every metric in this chapter, a definition rather than a fact — and the securities regulator treats it that way.

In January 2020 the U.S. Securities and Exchange Commission issued interpretive guidance on the use of key performance indicators and metrics in management's discussion and analysis. A company presenting a metric, the Commission said, should ordinarily accompany it with a clear definition of the metric and how it is calculated, a statement of why the metric is useful to investors, and a statement of how management uses it in managing the business (U.S. Securities and Exchange Commission, 2020). More striking, for this chapter's purposes, is what the guidance asks for when a definition changes. If a company alters the method by which a metric is calculated or presented, it should consider disclosing, to the extent material, the differences from prior periods, the reasons for the change, the effects of the change on both current and previously reported amounts, and any other methodological differences a reader would need in order to understand performance (U.S. Securities and Exchange Commission, 2020).

Read against Section 3.5.1, that is a regulator enforcing the same underlying discipline: define how the metric is calculated, explain why it matters and how management uses it, and disclose material changes in definition or presentation. The requirement exists because the alternative has been observed repeatedly, and the platform companies' own filings are unusually candid about it.

Facebook's quarterly filings carried a standing section titled "Limitations of Key Metrics and Other Data," disclosing that duplicate accounts may have represented approximately 10 percent of its worldwide monthly active users, and false accounts approximately 3 to 4 percent (Facebook, Inc., 2018). Those are not rounding errors; they are the numerator's known impurities, published because the metric could not be honestly read without them. Twitter's filings disclosed a different kind of gap. Its user counts included people reached through applications that "automatically contact our servers for regular updates with no discernible user-initiated action involved," a population the company quantified at fewer than 8.5 percent of monthly active users as of December 31, 2017 (Twitter, Inc., 2018). A reader who takes *active user* to mean a person who chose to open the application is not reading the same metric the company is reporting.

Neither case is an arithmetic mistake. Together they show two ways a user-count numerator can diverge from its ordinary-language meaning: known impurities that must be disclosed, and a technical inclusion rule that counts activity no person initiated. The practice lesson is that the four-element discipline is not a classroom formality. In public markets it is a disclosure obligation, and the specific thing regulators worry about is the situation this chapter opened with — one metric name denoting different quantities to different readers, or in different periods.

3.12.2 The Acquisition-Cost Dispute, and What Firms Build to End It

Inside companies the same problem arrives without a regulator. The most reliably contested marketing metric is customer acquisition cost, and the dispute follows a pattern recognizable from Section 3.1. Marketing computes it as campaign spend divided by new customers attributed to campaigns. Finance computes it as total sales and marketing expense — salaries, agency retainers, tooling, and all — divided by every new customer, attributed or not. Marketing's number is usually smaller — sometimes substantially so — and neither team has necessarily made an error. They have chosen different numerators, different denominators, and frequently different windows, because they are answering different questions: marketing is asking what a campaign returns, finance is asking what growth costs the company.

The dispute is survivable once. It becomes expensive when it recurs every quarter, because each recurrence spends senior meeting time re-litigating vocabulary. Two organizational responses are designed to prevent the dispute from recurring, and both are a data dictionary at institutional scale.

The first is technical. A semantic or metrics layer places the definition of each metric in one version-controlled location between the data warehouse and every tool that queries it, so that a dashboard, a notebook, and a spreadsheet all resolve "customer acquisition cost" to the same expression. The tooling in this space is explicit about the problem it exists to solve: multiple analysts each writing their own query for the same metric produces confusion, inconsistency, and an ongoing data-management burden (dbt Labs, n.d.). The layer does not decide what customer acquisition cost should mean. It makes the chosen meaning the path of least resistance, which in practice is most of the battle.

The second response is governance. Firms establish data governance councils, or appoint data stewards, with explicit authority over definitions. Khatri and Brown (2010) frame governance as a matter of decision rights rather than rules: the design question is who decides what, across domains including data principles, data quality, and metadata — and metadata is precisely where "what does this column mean" formally lives. A metric council does on a standing basis what the CFO did once in the opening case: it names an owner for each contested metric, records the definition, and adjudicates proposed changes.

The practice lesson is a distinction worth carrying into a first job. A data dictionary maintained by one conscientious analyst is a personal habit, and it leaves when that analyst does. A data dictionary with named owners, a review process, and a system that enforces it is an organizational capability. Only the second one stops the meeting from happening again next quarter.

3.12.3 The Attribution Window Is a Budget Decision

The third example is the one most likely to appear in a first analyst job, because it arrives disguised as a settings page. Advertising platforms count a conversion as caused by an ad if the conversion occurs within a stated period after the interaction. That period is the conversion, or attribution, window, and it is a parameter the advertiser sets.

Google Ads documents the arithmetic plainly. It sets a default 30-day click-through window, adjustable between 1 and 90 days depending on the conversion source, and a default one-day view-through window; for web conversions the engaged-view default is three days, while some app conversion types use shorter defaults (Google, n.d.). The consequence is stated without euphemism: a shorter conversion window reduces the number of conversions the account records for that conversion action, and conversions occurring after the window closes simply do not appear in reports (Google, n.d.). Nothing about the customers changes. The same people buy the same products on the same days. What changes is the window in the metric's definition — the third of the four elements from Section 3.5.1.

The consequence is budgetary, because reported conversions drive reported cost per acquisition and return on ad spend, and those figures drive allocation. A campaign whose customers deliberate for three weeks looks efficient under a 30-day window and wasteful under a 7-day one. If two channels are compared under different windows — which happens easily, since every platform ships its own defaults — the comparison is not close to fair, and the budget moves anyway. Attribution has a serious research literature precisely because these rules determine credit and therefore money; Berman (2018) shows that attribution rules do not merely describe advertising outcomes but change how advertisers and publishers behave.

The practice lesson is that some of the most consequential measurement decisions an analyst will ever make are exposed as configuration. They do not look like decisions, which is exactly why they are made carelessly. Recording the window alongside every attributed figure, and refusing to compare channels measured on different windows, is a small habit with a large budget attached.

3.12.4 In Your First Analyst Job

Across these three situations, one failure recurs, and it is not computational. A name outlives the definition it was given. The regulator's concern, the finance dispute, and the attribution window are versions of the same event: a phrase becomes load-bearing before anyone writes down what it means.

In your first analyst job you will inherit metric names. A few will be documented; most will not be. The productive first move is small and unglamorous. For any metric you are asked to produce, find its numerator, its denominator, its window, and its filters — and if nobody in the organization can state all four, you have just discovered the most valuable thing you will do that month. Write them down where the people who disagree can see them and object.

There is a reason to do this even when no one asks. In every organization, someone ends up being the person who knows what the numbers mean, and that person gets consulted before decisions are made rather than after. Whoever controls the definitions controls the conversation. The way to control them is not to guard them; it is to write them down.

3.13 Ethics, Privacy, and Responsible Measurement

So far this chapter has treated measurement as a matter of precision. This section treats it as a matter of responsibility, extending the discussions in Sections 1.13 and 2.12 from the use of data and the framing of questions to the design of measures themselves.

The first issue is that measurement allocates attention. What an organization measures, it manages; what it leaves unmeasured tends to become invisible in decisions. StyleCraft's tables measure purchases, clicks, and enrollments; they do not measure customer trust, employee strain in the new stores, or the experience of the customer who receives no offers because no model scores her as valuable. The analyst cannot measure everything, but the analyst can say out loud what a measurement scheme omits — the same representational duty that Section 2.12 assigned to problem framing.

The second issue is that proxies encode assumptions, and flawed proxies can systematically disadvantage groups of people. The healthcare-cost example of Section 3.6.1 is the canonical demonstration (Obermeyer et al., 2019), and the mechanism transfers directly to marketing: if "customer value" is operationalized as historical spend, then customers the company historically underserved — the markets where it barely marketed, the channels it launched late — will measure as low-value, and a targeting rule built on the proxy will underserve them again. The StyleCraft data contains a deliberate instance of this trap: the newest stores' customers look unresponsive partly because the stores are new and were under-marketed during ramp-up, and a naive measure of market potential would recommend abandoning them (Barocas & Selbst, 2016). Chapters 6 and 9 return to this artifact; the defense begins here, with the habit of asking what a proxy assumes.

The third issue is privacy as a measurement design choice. Every column is a decision to record something about a person, and granularity is part of the decision: StyleCraft stores `age_band` rather than birthdate and permits gender to be Undisclosed — small design choices that trade analytical precision for reduced exposure, in the spirit of collecting the minimum the purpose requires (Martin & Murphy, 2017). The data dictionary is where such choices become visible and reviewable: a dictionary that records what is collected, from where, and why is also, quietly, a privacy inventory. Furthermore, the same restraint applies to the analyst's tools: pasting customer-level data into an external AI assistant is a measurement-custody decision, not merely a convenience, and Section 1.13's warning applies with full force.

CONCEPT

Responsible Measurement

Responsible measurement is the practice of designing and documenting measures with awareness of what they omit, what their proxies assume, whom their errors would disadvantage, and what their collection costs the people being measured. It extends responsible analytics (Section 1.13) and responsible framing (Section 2.12) to the level of the variable itself.

Source: Adapted from Obermeyer et al. (2019), Barocas and Selbst (2016), and Martin and Murphy (2017).

3.14 Chapter Summary

This chapter examined the measurement foundations beneath every analysis in this guide. The main point is that a number is a claim: every reported figure presupposes a unit of analysis, an operational definition, a time window, and filters, and when these are left unstated, one metric name can honestly denote several quantities — the situation the StyleCraft meeting demonstrated with three repeat purchase rates of 60, 43, and 20 percent.

The chapter built the vocabulary for stating the claim precisely. Measurement translates concepts into columns through operational definitions. The grain of a table and the unit of analysis of a claim are declared separately and usually, but not necessarily, coincide; a data family contains fact tables recording measures and dimension tables supplying attributes, and moving between grains changes the claim, with the ecological fallacy waiting for analysts who forget. Variables carry different kinds of information — nominal, ordinal, interval, ratio — and the level of measurement, which stored data types do not record under any pandas version, governs permissible summaries, encodings, and charts. One business concept can appear as a metric, a KPI, a dimension, or a feature, and a ratio metric is defined only when its numerator, denominator, window, and filters are written down. Measures can fail by invalidity, unreliability, or poor data quality, and proxies deserve particular scrutiny. The chapter's central instrument, the data dictionary, records all of this — grains, columns, and metric definitions with their owners and decision uses — so that a team's data can mean the same thing to everyone, including the AI assistants the team supervises.

The lab reconstructed the three-number dispute from eighteen visible rows, reconciled every stored type against its intended measurement level, verified each figure by hand, and resolved the dispute by documentation rather than recomputation. The chapter then stepped outside StyleCraft to see the same mechanism at other scales: a securities regulator requiring companies to define their metrics and disclose definitional changes, a marketing-versus-finance dispute over acquisition cost that firms answer with semantic layers and governance councils, and an advertising conversion window that moves budget without changing a single customer's behavior.

Looking ahead, the dictionary describes the data as it should be; the files as they arrive are another matter. The next chapter opens StyleCraft's raw transaction data — with its missing values, duplicates, outliers, impossible dates, and inconsistent labels — and develops the discipline of data preparation and exploratory analysis: finding the defects Table 3.4 named, remedying them defensibly, and verifying an AI assistant's cleaning code before trusting a single row it touched.

3.15 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: name the unit of analysis, classify the variable, assign the role, define the metric completely, judge validity and quality, and document the result. They are organized into three groups. Core chapter practice is the required path and should be completed by every student. In-class activities are designed for discussion. Extensions are optional and go beyond the required material. Each exercise also carries its assignment label (required practice, in-class discussion, homework submission, or optional) so that instructors can assign selectively.

3.15.1 Core Chapter Practice

Exercise 3.1 Concept Check (Required Practice)

1. Explain the difference between a concept and an operational definition, using “active customer” as the example.
2. What is the grain of a table, and why does dimensional modeling treat declaring it as the first design decision?
3. How does the grain of a table differ from the unit of analysis of a claim, and give one situation in which the two come apart.
4. What distinguishes a fact table from a dimension table, and why is StyleCraft’s campaigns table an awkward case for that distinction?
5. State the four levels of measurement and give one StyleCraft column at each level that is not used as an example in Table 3.2.
6. Why can one field serve as a dimension in a dashboard and as a feature in a model? Give an example of each use for acquisition_channel.
7. What four elements define a ratio metric completely, and which of the four did the three StyleCraft teams disagree about?
8. Explain why reliability is necessary but not sufficient for validity, with a marketing example.

Exercise 3.2 Identify the Unit of Analysis (Required Practice)

For each question or dataset, name the unit of analysis of the claim, the grain of the table you would start from, and which StyleCraft base or derived table that is. Where the two differ, say what aggregation closes the gap.

1. What share of customers acquired through TikTok place a second order within 90 days?
2. What is the average discount depth on occasionwear lines?
3. Which store had the largest year-over-year revenue growth?
4. Did daily revenue accelerate after the Beverly Hills opening?
5. Which campaigns produced the lowest cost per click?
6. Is average order value higher for app orders than for online orders?
7. A file in which one row records one customer’s recency, frequency, and total spend.
8. A file in which one row records one day’s total revenue and marketing spend.

Exercise 3.3 Classify the Variables (Required Practice)

For each StyleCraft column, state the level of measurement (nominal, ordinal, interval, or ratio), the data type you would expect pandas to report on first load, and one operation that would be arithmetically possible but conceptually impermissible.

1. store_name
2. loyalty_tier
3. order_date
4. quantity
5. customer_id
6. age_band
7. discount_pct
8. is_occasionwear

Exercise 3.4 Metric, KPI, Dimension, or Feature? (Required Practice)

Classify each item in its described use as a metric, a KPI, a dimension, or a feature, and justify the classification in one sentence.

1. Average order value, computed for a one-time comparison of store types.
2. Repeat purchase rate, with a 45 percent target, owned by the CRM manager, reviewed monthly by the executive team.

3. store_type, used to break out revenue in a Tableau dashboard.
4. Days since last purchase, computed per customer as an input to the Chapter 9 churn model.
5. Email opt-in rate, reported once in a memo about signup-flow changes.
6. generation (Gen Z versus Gen Y), used to filter a campaign performance report.
7. Discount share of a customer's purchases, standardized and fed to a clustering algorithm.
8. Return on ad spend, on the CMO's quarterly scorecard with a target of 4.0.

Exercise 3.5 Define the Metric Precisely (Homework Submission)

Each phrase below circulates at StyleCraft without a written definition. For each, write a complete metric definition — numerator, denominator, window, filters, and grain — and then write a second, different but equally defensible definition. Conclude each pair with one sentence on which definition you would recommend, for what decision, and who should own it.

1. "Active customer."
2. "Conversion rate" for the mobile app.
3. "Average discount."
4. "Customer growth" for the quarterly board deck.
5. "Loyalty engagement."

Exercise 3.6 Hands-On Colab Practice (Homework Submission)

1. Open a new Google Colab notebook and reproduce Lab 3.1. Record the pandas version your environment reports and state whether your text columns arrived as object or str.
2. Add a Backstage-tier customer to the preview and rerun Code 3.2. Verify that max() and the "Insider or above" count respond as predicted.
3. Compute tenure_days for each customer as the days between signup_date and the analysis date (June 30, 2026). State the level of measurement of signup_date and of tenure_days, and explain why they differ.
4. Recompute Definition A with the denominator reindexed against the full customers table, so that customers with zero orders count as non-repeaters. Explain in two sentences why this preview gives the same answer and why a real CRM would not.
5. Write a fourth repeat-rate definition: trailing 12 months, all channels, but restricted to customers who signed up before the window opened. Predict whether it will be higher or lower than Definition B, compute it, and explain the result in two sentences.
6. Add your fourth definition to the metric_dictionary frame with every element completed, including an owner and the decision it would inform.
7. Report one verification check you performed and one way this ten-customer preview could mislead if treated as representative.

Exercise 3.7 AI-Assisted Practice (Homework Submission)

1. Paste the customers preview from Lab 3.1 into an AI assistant and use the prompt from the AI in Practice box in Section 3.9 to draft dictionary entries for all six columns.
2. Audit the draft: identify at least one allowed-values range the assistant inferred from the sample rather than from business rules, and at least one known issue it could not have known.
3. Ask the assistant to define "repeat purchase rate" with no further context, and compare its answer with the three definitions in Table 3.6. Which did it choose, and did it state that it was choosing?
4. Document the exchange using the AI-use documentation template in Appendix D (introduced as Table 1.6).

Exercise 3.8 Build the Data Dictionary (Homework Submission)

Using the format of Table 3.5, write complete dictionary entries for all seven columns of the StyleCraft products table (Table 3.1 lists the table; the schema appears in the companion repository). For each column, include a plausible known issue. Then add one metric entry, in the format of Table 3.6 including owner and decision use, for "average selling price," and explain in two sentences how it differs from the unit_price column.

Exercise 3.9 Managerial Memo Exercise (Homework Submission)

Write a memo of 250–350 words from the analytics team to the StyleCraft CMO and CFO, dated before the October 15 expansion review, proposing the official definition of repeat purchase rate. The memo must state the decision the metric will inform, the recommended definition with all four elements explicit, the two rejected definitions and the legitimate uses each retains, the owner and review cadence, one validity limitation of the metric as a proxy for loyalty, and one anticipated gaming behavior with a countermeasure. Recommended structure: decision context, recommended definition, treatment of alternatives, ownership, limitations and risks.

3.15.2 In-Class Activities

Exercise 3.10 Find the Flaw in the Measurement (In-Class Discussion)

1. The averaged identifier. An AI-generated summary of the transactions table includes a row reporting the mean of customer_id. What went wrong, and at which stage of this chapter's checks would it have been caught?
2. The alphabetical tiers. A bar chart shows customer counts by loyalty_tier in the order Backstage, Insider, None, VIP, and a stakeholder concludes the program "peaks in the middle." Diagnose the error.
3. The generous window. A team reports that "78 percent of customers are repeat buyers" using the lifetime definition, in a deck about the current quarter's retention. Is the number wrong? What is it, precisely?
4. The proxy promotion. To raise "loyalty," measured by tier enrollment, stores begin auto-enrolling every checkout. Enrollment doubles. Has loyalty doubled? Connect your answer to Sections 2.12 and 3.6.
5. The grain jump. A slide states: "Stores with higher average order value have lower repeat rates, so we should discourage large baskets." Identify the fallacy and restate what the evidence supports.
6. The two nulls. An analyst drops all rows with missing campaign_id before computing campaign ROAS. What has the analyst assumed about the nulls, and what is the alternative?
7. The moving window. An analyst filters orders with a one-sided condition, keeping everything on or after the window start. The dashboard is scheduled to refresh weekly. What happens to the reported rate over the following three months, and why is the trend an artifact?

Exercise 3.11 Ethics of Measurement Mini-Cases (In-Class Discussion)

For each case, identify the measurement issue, the stakeholders affected, and what a responsible analyst should do instead.

1. The convenient proxy. A "customer value" score built on historical spend is used to allocate service priority. The lowest scores cluster in the markets where StyleCraft opened most recently and marketed least.
2. The optional field. To improve targeting, a proposal would make age and gender required at signup rather than optional. Conversion on the signup form is expected to drop, and the fields would be stored at full precision.
3. The unmeasured cost. The expansion review measures store revenue, cost, and repeat rate, but nothing about the staff turnover in the new stores, which the retail team privately reports is severe.

3.15.3 Extensions

Exercise 3.12 Reflection Questions (Optional)

Finally, the following questions are intended to be thought-provoking rather than graded.

1. Think of a metric that matters in an organization you know. Can you state its numerator, denominator, window, and filters? If not, who could?
2. Have you ever seen a meeting stall over two versions of "the same" number? In this chapter's terms, what was actually in dispute?
3. Which marketing concept do you believe is measured worst by the proxies commonly used for it, and what would a better measure require?

4. What information about you, as a customer, do you suspect is recorded at finer granularity than the purpose requires?
5. If a data dictionary prevents disputes, why do so few organizations maintain one? What would make the habit stick?

3.16 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Data dictionary. A shared, maintained document recording, for each table, its grain; for each column, its definition, type, level of measurement, allowed values, source, and known issues; and, in the extended form used in this guide, for each named metric, its numerator, denominator, window, filters, grain, owner, and the decision it informs (adapted from Kimball & Ross, 2013).

Data quality. The degree to which data is fit for its intended uses, assessed along dimensions including accuracy, completeness, consistency, uniqueness, timeliness, and conformity (course synthesis informed by Wang & Strong, 1996; Redman, 1998).

Dimension. An attribute used to slice, group, or filter metrics in descriptive analysis and dashboards, typically drawn from a dimension table (Kimball & Ross, 2013).

Dimension table. A table whose rows describe entities — customers, products, stores — and whose columns supply the attributes by which facts are sliced, grouped, and filtered (Kimball & Ross, 2013).

Ecological fallacy. The error of inferring individual-level relationships from group-level data (Robinson, 1950).

Fact table. A table whose rows record events or measurements at a declared grain, typically carrying quantitative measures, many of which can be aggregated to construct metrics; StyleCraft's transactions table is its primary sales fact table (Kimball & Ross, 2013).

Feature. A variable prepared as an input to a predictive model, often engineered from raw columns at the model's unit of analysis (Provost & Fawcett, 2013; Domingos, 2012).

Grain. What one row of a table represents. Grain is a property of a table and is declared when the table is designed; it usually, but not necessarily, coincides with the unit of analysis of the claims computed from it (Kimball & Ross, 2013).

Key performance indicator (KPI). A metric elevated to guide action: connected to strategy, attached to a target, assigned an owner, and reviewed on a cadence (Parmenter, 2015).

Level of measurement. The classification of a variable as nominal, ordinal, interval, or ratio, which governs the mathematical operations and summaries the variable supports (Stevens, 1946).

Measurement. The assignment of numbers, labels, or categories to observations according to explicit rules (Stevens, 1946).

Metric. A quantity computed from data by an explicit rule for description or comparison; a ratio metric is fully specified by its numerator, denominator, window, and filters (adapted from Farris et al., 2010).

Operational definition. The specific written rule by which a concept is turned into a recorded or computed value, including population, window, and filters (adapted from Churchill, 1979; Carmines & Zeller, 1979).

Proxy measure. A measure adopted to stand in for a concept that cannot be observed directly; proxies import assumptions that can fail silently or inequitably (Obermeyer et al., 2019).

Reliability. The degree to which a measure produces consistent results under repeated or equivalent measurement (Carmines & Zeller, 1979).

Responsible measurement. The practice of designing and documenting measures with awareness of what they omit, what their proxies assume, whom their errors would disadvantage, and what their collection costs the people measured (adapted from Obermeyer et al., 2019; Barocas & Selbst, 2016; Martin & Murphy, 2017).

Stored data type. The type a software system uses to hold a column's values — integer, decimal, Boolean, datetime, or text. The stored type records how a value is kept, not what level of measurement it carries, and its name may change between software versions (pandas development team, n.d.).

Unit of analysis. The entity about which an analysis makes its claim, which the analytic specification requires to be named (Section 2.5). Distinct from, though usually aligned with, the grain of the table the claim is computed from (Kimball & Ross, 2013; Robinson, 1950).

Validity. The degree to which a measure captures the concept it is intended to capture (Carmines & Zeller, 1979; Churchill, 1979).

Variable. Any characteristic recorded for each unit in a dataset; a column considered as a carrier of information rather than as storage (Stevens, 1946).

3.17 Further Readings

Students who want additional background may begin with the following readings. Measurement-focused sources are listed first because this chapter approaches analytics from the measurement side.

- Stevens (1946) for the original statement of the four levels of measurement, still short and readable.
- Churchill (1979) for the classic paradigm for developing marketing measures, and for the argument that poor translation of constructs, not poor statistics, is the usual culprit.
- Kimball and Ross (2013) for grain, fact tables, and dimensions — the data-modeling vocabulary used by analytics teams everywhere.
- Robinson (1950) for the ecological fallacy in its original form.
- Wang and Strong (1996) for the consumer-centered view of data quality, and why "good data" is always relative to a use.
- Obermeyer et al. (2019) for the consequences of a flawed proxy deployed at scale.

3.18 References

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Berman, R. (2018). Beyond the last touch: Attribution in online advertising. *Marketing Science, 37*(5), 771–792. https://doi.org/10.1287/mksc.2018.1104

Campbell, D. T. (1979). Assessing the impact of planned social change. *Evaluation and Program Planning, 2*(1), 67–90. https://doi.org/10.1016/0149-7189(79)90048-X

Carmines, E. G., & Zeller, R. A. (1979). *Reliability and validity assessment*. Sage.

Churchill, G. A., Jr. (1979). A paradigm for developing better measures of marketing constructs. *Journal of Marketing Research, 16*(1), 64–73. https://doi.org/10.1177/002224377901600110

dbt Labs. (n.d.). *About MetricFlow*. dbt Developer Hub. Retrieved July 27, 2026, from https://docs.getdbt.com/docs/build/about-metricflow

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Facebook, Inc. (2018). *Quarterly report (Form 10-Q) for the quarterly period ended March 31, 2018*. U.S. Securities and Exchange Commission. https://www.sec.gov/Archives/edgar/data/1326801/000132680118000032/fb-03312018x10q.htm

Farris, P. W., Bendle, N. T., Pfeifer, P. E., & Reibstein, D. J. (2010). *Marketing metrics: The definitive guide to measuring marketing performance* (2nd ed.). Pearson Education.

Google. (n.d.). *About conversion windows*. Google Ads Help. Retrieved July 27, 2026, from https://support.google.com/google-ads/answer/3123169

Khatri, V., & Brown, C. V. (2010). Designing data governance. *Communications of the ACM, 53*(1), 148–152. https://doi.org/10.1145/1629175.1629210

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

Martin, K. D., & Murphy, P. E. (2017). The role of data privacy in marketing. *Journal of the Academy of Marketing Science, 45*, 135–155. https://doi.org/10.1007/s11747-016-0495-4

Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science, 366*(6464), 447–453. https://doi.org/10.1126/science.aax2342

pandas development team. (n.d.). *Migration guide for the new string data type (pandas 3.0)*. pandas documentation. Retrieved July 27, 2026, from https://pandas.pydata.org/docs/user_guide/migration-3-strings.html

Parmenter, D. (2015). *Key performance indicators: Developing, implementing, and using winning KPIs* (3rd ed.). Wiley.

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. *Big Data, 1*(1), 51–59. https://doi.org/10.1089/big.2013.1508

Redman, T. C. (1998). The impact of poor data quality on the typical enterprise. *Communications of the ACM, 41*(2), 79–82. https://doi.org/10.1145/269012.269025

Robinson, W. S. (1950). Ecological correlations and the behavior of individuals. *American Sociological Review, 15*(3), 351–357. https://doi.org/10.2307/2087176

Stevens, S. S. (1946). On the theory of scales of measurement. *Science, 103*(2684), 677–680. https://doi.org/10.1126/science.103.2684.677

Strathern, M. (1997). “Improving ratings”: Audit in the British university system. *European Review, 5*(3), 305–321. https://doi.org/10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4

Twitter, Inc. (2018). *Annual report (Form 10-K) for the fiscal year ended December 31, 2017*. U.S. Securities and Exchange Commission. https://www.sec.gov/Archives/edgar/data/1418091/000156459018003046/twtr-10k_20171231.htm

U.S. Securities and Exchange Commission. (2020). *Commission guidance on management’s discussion and analysis of financial condition and results of operations* (Release Nos. 33-10751; 34-88094). https://www.sec.gov/rules/interp/2020/33-10751.pdf

Velleman, P. F., & Wilkinson, L. (1993). Nominal, ordinal, interval, and ratio typologies are misleading. *The American Statistician, 47*(1), 65–72. https://doi.org/10.1080/00031305.1993.10475938

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems, 12*(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Chapter 04

Data Preparation and Exploratory Analysis

From Raw Files to Trustworthy Tables: Cleaning, Verification, and First Exploration

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops data preparation and exploratory analysis as connected disciplines. It presents preparation as a four-stage pipeline — raw, inspected, cleaned, documented — and introduces the chapter's central instrument, the verification log, in which the analyst predicts each cleaning step's effect on row counts and totals before running the step and records the actual values afterward. The chapter treats the major defect families in turn: missing values and the bias each remedy introduces, legitimate extremes against genuine errors, exact and near-duplicate records, and inconsistencies of type, format, and category. It covers the constructive side of preparation — derived variables and deliberate grain changes — and introduces exploratory data analysis as disciplined questioning. A section on AI assistance names the characteristic failure modes of delegated cleaning code. The labs clean StyleCraft's transactions_raw file against its documented defect catalog and certify the result against the transactions_clean answer key.

KEYWORDS

data preparation; data cleaning; verification log; missing data; imputation; outliers; duplicate records; type coercion; exploratory data analysis; marketing analytics

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Data preparation and exploratory analysis. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 4, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Matplotlib is a NumFOCUS fiscally sponsored project. dbt is a trademark of dbt Labs, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the

underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

6. Describe the preparation pipeline — raw, inspected, cleaned, documented — and explain why cleaning decisions are analytic decisions rather than mechanical chores.
7. Direct an AI assistant to load, inspect, and transform a dataset in Google Colab, and verify the results against predictions recorded in a verification log.
8. Run and interpret the standard inspection routines — shape, data types, head, info, and describe — as the first contact with any new file.
9. Distinguish the three missing-data mechanisms at an intuitive level, choose among dropping, imputing, and flagging, and state the bias each remedy introduces.
10. Identify and handle outliers using interquartile-range fences, z-scores, and domain limits, and defend the distinction between a legitimate extreme and a genuine error.
11. Detect and remove exact and near-duplicate records, and explain why near-duplicates evade the obvious check.
12. Coerce types and standardize formats and category labels, including dates, currency strings, and inconsistent categorical values.
13. Create derived variables and grouped summaries, including deliberate changes of grain, and verify that an aggregation preserves the totals it should preserve.
14. Produce and interpret exploratory visualizations — histograms, box plots, bar charts, and scatterplots — reading them for anomalies, distribution shape, and questions worth pursuing.
15. Recognize the characteristic failure modes of AI-generated cleaning code and audit such code before trusting any row it touched.

Chapter 3 closed with a warning. The data dictionary describes the data as it should be; the files as they arrive are another matter. Chapter 3 supplied the vocabulary for that gap — the six data quality dimensions of Section 3.7 and the defect examples of Table 3.4 — and promised that the next chapter would do the repair. This chapter keeps the promise. It develops the discipline of data preparation: finding the defects a raw file contains, remedying them defensibly, documenting every choice, and verifying the result before a single downstream number is computed from it. It then turns to the first analytical use of a prepared table, exploratory data analysis, in which the analyst interrogates distributions and charts for surprises before committing to any formal method.

CONCEPT

What This Chapter Is Really About

Chapter 3 argued that a number is a claim. This chapter makes the uncomfortable companion argument: before any claim can be computed, someone edits the evidence. Every cleaning decision — this row is a duplicate, that value is an error, these labels are the same category, those records are out of scope — changes the numbers every later chapter will report, and most of these decisions are invisible in the final analysis. An undisciplined analyst makes them silently and forgets them; a disciplined analyst makes them explicitly, predicts their effect before executing them, records predicted against actual in a verification log, and documents every exclusion. The technical content of this chapter is pandas mechanics; the professional content is editorial accountability. In other words, cleaning is not what happens before the analysis. Cleaning is the first analysis.

4.1 Marketing Decision Context: The File That Does Not Add Up

Chapter 3 ended with a small institutional victory: StyleCraft adopted a headline definition of repeat purchase rate, assigned it an owner, and published a data dictionary. The dictionary now records what every column in the company's data family is supposed to mean. Ten days before the quarterly expansion review, the analytics team learns what the columns actually contain.

The setting is the review itself. The Chapter 2 memo committed StyleCraft to revisiting the paused store openings this quarter, with two named analyses in hand, and both analyses — the same-age cohort comparison and the store-economics summary — must be computed from transaction data. To support them, the data engineering contractor has assembled a consolidated extract, transactions_raw, merging order lines from the e-commerce platform, the point-of-sale system in the stores, and the mobile app into a single file at the order-line grain described in Table 3.1. The extract covers the full twenty-four-month history. It is the file from which every number in the review will be computed.

The trouble begins with a routine reconciliation. Before building anything, the analytics lead compares the extract's total revenue for the last complete quarter against the figure the finance system reported for the same quarter — a known total, produced by an audited system, for exactly the period the extract should match. The two figures disagree. The extract is higher, by a margin of a few percentage points: small enough that a hurried analyst might shrug it off, and large enough, on a quarterly revenue base, to change the story the expansion numbers tell. A quick inspection makes matters worse rather than better. Some rows show a quantity of 999. Some prices sit far outside the catalog's $24-to-$90 band, and others arrive not as numbers at all but as text with a currency symbol. The channel column contains “Online,” “online,” and “web,” which the group-by from Chapter 3's lab would happily count as three channels. A spot check against the customers table turns up order dates earlier than the ordering customer's signup date. Furthermore, several blocks of rows look suspiciously like other blocks of rows.

The CFO's question from Chapter 3 — “Which of these numbers is wrong?” — returns in a sharper form. This time the question is not which definition to prefer, but whether the underlying records can be trusted at all, and the deadline is fixed: the review convenes in ten days, and every analysis on its agenda inherits whatever this file contains. The decision the analytics lead must make is therefore not any single cleaning choice. It is whether, and on what documented basis, to certify a cleaned version of transactions_raw as the table of record for the expansion review — knowing that the CFO will ask not only what the numbers are, but what was removed, what was changed, and how the team knows the cleaned file reconciles to finance's known total.

This chapter is the ten days. Sections 4.2 and 4.3 establish the pipeline and the first inspection. Sections 4.4 through 4.7 treat the defect families one at a time — missing values, outliers, duplicates, and inconsistent types and labels — because each family demands a different diagnosis and carries a different risk. Sections 4.8 and 4.9 turn from repairing the file to using it: deriving variables, changing grain deliberately, and exploring the cleaned data for the surprises that formal analysis should know about in advance. Section 4.10 examines how AI assistants help and fail at exactly this work, and the labs in Section 4.11 perform the certification on StyleCraft's real files, verification log and all.

4.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The extract's revenue exceeds finance's known total. Which of the defects listed above could inflate revenue, which could deflate it, and which could do either?
- The analytics lead reconciled against a known total before doing anything else. What other known quantities could serve the same anchoring role for this file?
- If the team simply deleted every suspicious row, the file would reconcile eventually. Why is that not the same as certifying the file?
- Who, besides the analytics team, needs to be able to reconstruct what was removed and why — and what document should make that possible?

4.2 The Preparation Pipeline: Raw, Inspected, Cleaned, Documented

The opening case presents a file that cannot be trusted and a deadline that does not care. The temptation in such moments is to start fixing things — delete the 999s, dedupe, move on. This section argues for a slower first move: adopting a pipeline that separates the stages of preparation and preserves the evidence at every stage.

Data preparation, as this guide uses the term, is the disciplined transformation of raw source data into a documented, analysis-ready table, through the repeatable sequence raw → inspected → cleaned → documented. Each stage has a distinct obligation. The raw stage preserves the file exactly as it arrived; the raw file is never edited in place, because it is the evidence against which every later claim of "we fixed X" is checked. The inspected stage produces a written account of what the file contains — its shape, types, ranges, and apparent defects — before anything is changed, so that remedies respond to diagnoses rather than to habit. The cleaned stage applies remedies one defect family at a time, in code, in an order that is recorded and can be re-run from the untouched raw file. The documented stage records what was done and why: which rows were removed and under what rule, which values were changed or filled, which decisions were judgment calls, and how the result was verified. The practitioner literature converges on the same broad sequence and on a durable practical observation: preparation often consumes a larger share of analytical work than modeling itself (Dasu & Johnson, 2003; Anaconda, 2020).

> **DEFINITION**
>
> **Data Preparation**
>
> Data preparation is the disciplined transformation of raw source data into an analysis-ready table through a repeatable, documented sequence of inspection, cleaning, and verification steps, such that the resulting table can be reproduced from the untouched raw file and every change to the data can be traced to a recorded decision.
>
> Source: Adapted from Dasu and Johnson (2003) and Hellerstein (2008).

Two properties of the pipeline deserve emphasis, because they distinguish professional preparation from improvised tidying. The first is reproducibility. Because every remedy is a step in code applied to the preserved raw file, the entire cleaned table can be regenerated at will — after a bug is found, after a rule is revised, after a stakeholder asks "what happens if we keep those rows?" An analyst who cleaned by hand-editing a spreadsheet can answer none of these. In code, reproducibility has a concrete first line: the raw object is loaded once and immediately copied, and every transformation is applied to the copy, so that the baseline against which the log's predictions are checked is still in memory at the end of the pipeline. The labs in Section 4.11 make that copy the first executable step, and Exercise 4.7 shows what goes wrong when it is skipped.

The second property is order-awareness. Cleaning steps interact: deduplicating before fixing the currency-string prices means the duplicate check compared text to text; filling missing values before removing duplicates means some fills were computed from rows about to be deleted. There is no single correct order for every file, but there is always a chosen order, and the choice belongs in the documentation. As a default, this guide inspects first, fixes types and labels second (so that later comparisons compare like with like), removes duplicates third, resolves impossible values and outliers fourth, treats missing values fifth, and derives new variables last, verifying after every step.

4.2.1 The Verification Log

The previous paragraphs used the word "verify" repeatedly without saying how. This subsection introduces the chapter's central instrument, which is also this chapter's expression of the predict-then-verify discipline from Section 1.7: the verification log.

A verification log is a running record, kept alongside the cleaning code, in which the analyst writes down — before executing each step — what the step is expected to do to the file's checkable quantities, and then records what actually happened. The checkable quantities are deliberately humble: the row count, the count of distinct orders and customers, the sum of revenue, the count of missing values in the affected columns. If the plan is to remove exact

duplicate order lines, the log first records the number of exact duplicates found during inspection and the predicted row count after removal; the step is then run, the actual count is recorded, and any gap between prediction and outcome must be explained before the pipeline proceeds. A mismatch is not an annoyance; it is the log working. Every mismatch means the analyst's model of the data or the analyst's model of the code is wrong, and both are exactly the things worth knowing before the expansion review, not after.

The log earns its keep twice more downstream. At certification time, it is the answer to the CFO's question: the sequence of entries, read top to bottom, is a complete account of how the raw file became the table of record, including the reconciliation against finance's known total. Certification is a threshold, not a mood — a file is certified for a given use when every step's prediction has been confirmed or explained and no quantity the use depends on remains unresolved, and Section 4.11 writes that threshold as a line of code rather than a sentence of reassurance. And whenever an AI assistant writes the cleaning code — the normal case in this course — the log is the audit that makes delegation safe, a point Section 4.10 develops.

CONCEPT

The Verification Log

For every cleaning step, record four things before running it: the step's rule in one sentence; the quantities it should change (rows, totals, missing counts) and by how much; the quantities it should leave untouched. Run the step; record the actual values; explain any difference before continuing. A cleaned file is certified for a stated use when the log runs from the raw file's baseline to the final table with every step's prediction either confirmed or explained, and no quantity that use depends on is left unresolved.

Source: Course concept developed for this guide, informed by Dasu and Johnson (2003) and Hellerstein (2008).

Table 4.1 summarizes the pipeline and shows where the log sits in it. With the frame established, the next section performs the pipeline's first stage on a new file: inspection.

Table 4.1

The preparation pipeline and its obligations

Stage	Obligation	Key artifact	Cardinal sin
Raw	Preserve the file exactly as it arrived	The untouched raw file	Editing the raw file in place
Inspected	Diagnose before treating: shape, types, ranges, defects	Written inspection notes; verification-log baseline	Fixing on sight without recording the diagnosis
Cleaned	Remedy one defect family at a time, in code, in a recorded order	The cleaning notebook; one log entry per step	Silent steps whose effects were never predicted
Documented	Make every change traceable and the result reproducible	The completed verification log; dictionary updates	A cleaned file whose differences from raw cannot be explained

4.3 First Contact: Inspection Routines

The pipeline's first active stage is inspection, and this section makes it concrete. The goal of inspection is a written diagnosis: what the file contains, how it is typed, what ranges its values occupy, and which of the defect families from Sections 4.4 through 4.7 appear to be present. Nothing is changed at this stage. In Chapter 3's editorial metaphor, this is the read-through before any correction is marked.

Five routines, all one-liners in pandas, do most of the work, and they should be run in roughly this order on any new file. The shape attribute reports rows and columns; the row count becomes the first line of the verification log, the baseline every later count is checked against. The dtypes attribute reports the stored type of every column, to be

compared — predict first, then look — against the types the data dictionary says each column should have; Chapter 3's lab showed how much measurement information the stored types can silently discard, and a date arriving as text or a price arriving as object is often the first visible symptom of the format problems Section 4.7 treats. The head method displays the first rows, which is where a human eye catches what summaries miss: a currency symbol in a numeric column, a label in the wrong case, a column that is obviously two facts packed into one. The info method combines shape, types, and — critically — the count of non-null values per column, turning missing data from a suspicion into a per-column tally. Finally, the describe method reports count, mean, standard deviation, minimum, quartiles, and maximum for numeric columns, and its extremes are the fastest defect detector available: a maximum quantity of 999 or a minimum price below the catalog floor announces itself in one line of output.

Two habits convert these routines from ritual into diagnosis. The first is prediction. Before running describe on unit_price, the analyst who has read the data dictionary knows the catalog band is $24 to $90 and writes that expectation down; the output is then read as a comparison, not a revelation. This is predict-then-verify (Section 1.7) applied at the cheapest possible point, before any code that changes anything has been written. The second is anchoring against known totals. Somewhere outside the file there is usually a number the file must agree with — finance's quarterly revenue, the loyalty platform's member count, last month's certified report. Computing the file's version of that number during inspection, as StyleCraft's analytics lead did, establishes at the outset how far from reconciled the raw file is, and gives the final log entry its target.

Inspection ends with a short written list: the defects observed, the columns affected, and the apparent scale of each problem. For StyleCraft's extract, that list will name missing values, outliers, duplicates, and inconsistent types and labels — the four families the next four sections treat in turn, beginning with the defect that is hardest to see because it is not there: the missing value.

4.4 Missing Values: Mechanisms, Remedies, and the Bias Each Introduces

Inspection produces, among other things, a per-column count of missing values. This section asks the two questions that count cannot answer: why are the values missing, and what should be done about them? The order of the questions matters. The remedy that is defensible depends on the mechanism that produced the absence, and analysts who reach for a remedy first are treating a symptom whose cause they have not diagnosed.

Begin with what Chapter 3 already established. Section 3.7 noted that in StyleCraft's transactions a missing campaign_id can mean two opposite things: the order was legitimately unattributable to any campaign (an informative null — the absence is the value), or the attribution pipeline failed (a true gap — the value existed and was lost). The two absences are indistinguishable in the file and opposite in meaning, and no amount of pandas can tell them apart; only knowledge of the source system can. The first move with any missing values is therefore not technical at all: consult the data dictionary (Section 3.8), whose known-issues field exists to record exactly this, and the system owners, who hold what no file contains.

For absences that are genuine gaps, the statistical literature distinguishes three mechanisms, and this guide needs them only at the level of intuition (Rubin, 1976; Little & Rubin, 2019). A value is missing completely at random (MCAR) when the fact that it is missing has nothing to do with anything — as if the recording system dropped values by coin flip. A value is missing at random (MAR) when the missingness is related to other observed columns but not to the missing value itself: if in-store customers skip the optional age question more often than online customers do, age_band is MAR given channel. A value is missing not at random (MNAR) when the missingness depends on the value that would have been recorded: if customers with the highest incomes are the most likely to decline an income question, the absences themselves are systematically hiding one end of the distribution. The names are awkward — MAR does not mean “random” in the everyday sense — but the escalation matters. Under MCAR, the observed rows are a smaller but unbiased sample. Under MAR, the observed rows are biased, and the bias can in principle be corrected using the observed columns. Under MNAR, the observed rows are biased in a way the data cannot itself reveal or repair.

DEFINITION

Missing-Data Mechanisms

The missing-data mechanism is the process by which absences arise in a dataset. Values are missing completely at random (MCAR) when missingness is unrelated to any variable; missing at random (MAR) when missingness is related to observed variables but not to the missing value itself; and missing not at random (MNAR) when missingness depends on the unobserved value. The mechanism, not the count of absences, determines which remedies are defensible and what bias they carry.

Source: Adapted from Rubin (1976) and Little and Rubin (2019).

Three families of remedy are available, and each purchases convenience with a specific bias that the analyst must be able to name. Dropping removes the affected rows (or, rarely, the affected column). It is simple, honest, and, under MCAR, costs only sample size. Under MAR or MNAR, however, dropping deletes a nonrandom slice of the world: if in-store shoppers disproportionately lack age_band, dropping those rows quietly removes in-store shoppers from every later analysis that uses age — precisely the customers the suburban expansion question is about. Imputing fills the gaps with estimated values — the column's mean or median, the most frequent category, or a value predicted from other columns. Imputation preserves rows, but it manufactures data: filling with the mean shrinks the column's variability and overstates confidence in later estimates, and filling categorical gaps with the most frequent label inflates the majority category (Little & Rubin, 2019; Osborne, 2013).

Flagging adds a companion indicator column — age_band_missing — so that the absence itself becomes analyzable, and it is normally combined with one of the other two rather than used alone. Flagging is the least glamorous remedy and frequently the most useful, because it preserves the one fact dropping and imputing both destroy: which rows were touched. It is not, however, free, and it is not a remedy in its own right. A flag does not resolve the missingness; it records it. It adds a column that every downstream reader must understand, and flagging every column indiscriminately produces a table full of indicators nobody interprets — and, in the predictive chapters ahead, features that can leak information about the outcome if the missingness is itself a consequence of it. The discipline is to flag where the absence is analytically interesting or where a judgment call needs to be testable later, and to say so in the log. Table 4.2 summarizes the trade.

Table 4.2

Missing-value remedies and the bias each introduces

Remedy	What it does	Defensible when	Bias it introduces
Drop rows	Removes rows with missing values in scoped columns	Mechanism plausibly MCAR; affected share small; rows not central to the question	Deletes a nonrandom slice under MAR/MNAR; shrinks sample
Impute (mean/median)	Fills numeric gaps with a central value	Numeric column; small gap share; downstream use tolerates damped variability	Understates spread; distorts distribution shape; fabricates precision
Impute (most frequent)	Fills categorical gaps with the modal label	Categorical column; gap share very small	Inflates the majority category; can flip group comparisons
Impute (model-based)	Predicts the missing value from other columns	MAR plausible; the imputation model is itself verified	Imports the model's assumptions; harder to audit
Flag	Adds an indicator column marking absence	Often useful for auditability, usually alongside a justified remedy	Does not alter observed values, but does not repair the missingness; the indicator must be interpreted and governed, and indiscriminate flagging can leak information into later models

Two closing disciplines connect this section back to the pipeline. First, every missing-value decision is a log entry: the count of absences before, the rule applied, the count after, and — for drops — the number of rows removed, predicted and then confirmed. Second, every decision is a documented exclusion or alteration in the sense Section 4.14 will examine ethically: filling in-store shoppers' ages with the online majority's median is an editorial act with consequences for whose behavior the analysis represents, and it belongs in writing. However the gaps are treated, the next defect family is easier to see and harder to explain away: values that are present but implausible.

4.5 Outliers: Legitimate Extremes and Genuine Errors

Missing values are absences; outliers are presences that strain belief. This section develops the chapter's second defect family, and it begins with the distinction on which every outlier decision turns: an outlier is a value that is far from the rest, and being far from the rest is not, by itself, evidence of being wrong.

An outlier is an observation that lies unusually far from the bulk of the data (Barnett & Lewis, 1994). The definition is deliberately about distance, not error, because extreme values arise in two ways that demand opposite treatment. Some extremes are legitimate: StyleCraft's occasionwear baskets are designed to be large, and a suburban customer buying four dresses at the top of the $24-to-$90 band produces a big — and real — order line. Deleting such rows does not clean the data; it amputates the phenomenon, and in StyleCraft's case it would amputate the exact behavior the expansion review needs to see. Other extremes are errors: a quantity of 999 in an apparel transaction is not a large purchase, it is a sentinel or entry error, and a unit price of $2,400 in a catalog that tops out near $90 is a decimal slip (Table 3.4 previewed both). The analyst's job is not to remove outliers; it is to classify extremes as legitimate or erroneous, using evidence, and then treat only the errors.

Detection comes first, and three rules cover most practice. The interquartile-range rule, inherited from Tukey's exploratory tradition, computes the first and third quartiles and flags values below $Q1 - 1.5 \times IQR$ or above $Q3 + 1.5 \times IQR$ (Tukey, 1977); it is robust, distribution-light, and it is the rule the box plot of Section 4.9 renders as a picture. The z-score rule flags values more than some threshold — conventionally three — standard deviations from the mean; it is familiar but fragile, because the outliers being hunted inflate the very mean and standard deviation used to hunt them, and it presumes a roughly symmetric distribution that marketing data rarely provides. Domain limits flag values outside ranges the business itself declares: prices outside the catalog band, discounts above the maximum the promotion engine permits, quantities beyond any plausible basket. Domain limits are the most powerful rule of the three, because they encode knowledge rather than statistics — and they come straight from the data dictionary's allowed-values field (Section 3.8), which is one more argument for writing that field down.

Classification and treatment follow detection. For each flagged value, the analyst seeks corroboration: does the extreme row make sense as a whole (a large line_revenue supported by a plausible quantity and an in-band price is a basket; a large line_revenue produced by quantity 999 is not)? Does the source system have a known sentinel convention? Does the same customer or store generate other extremes? Errors are then treated — corrected where the true value is recoverable (the $2,400 that is demonstrably $24.00), set to missing where it is not (converting the problem to Section 4.4's, with the mechanism known), or excluded with a documented rule — while legitimate extremes are kept, and kept visibly: the verification log records how many values each rule flagged, how each flagged value was classified, and what the treatment changed. Winsorizing — capping extremes at a percentile rather than removing them — appears in practice, but this guide treats it with caution: it alters real values wholesale, and in a dataset whose most decision-relevant behavior lives in the right tail, capping the tail is editing the story.

DEFINITION

Outlier

An outlier is an observation that lies unusually far from the bulk of the data, as judged by a stated rule such as the interquartile-range fences, a z-score threshold, or domain limits. An outlier is a candidate for investigation, not for removal: the analyst must classify it as a legitimate extreme, to be retained, or a genuine error, to be corrected, set to missing, or excluded under a documented rule.

Source: Adapted from Barnett and Lewis (1994) and Tukey (1977).

The discipline, restated: detect with a rule, classify with evidence, treat only errors, log everything. The next defect family requires no classification judgment at all — a duplicated order line is simply wrong — but it compensates by being harder to find than it looks.

4.6 Duplicate Records: Exact and Near

Outliers at least announce themselves in a describe output. Duplicates hide in plain sight: every individual value in a duplicated row is perfectly plausible, and the defect exists only in the relationship between rows. This section treats the two forms duplication takes and the reason the second form evades the first form's check.

An exact duplicate is a row identical to another row in every column — in a transaction file, the same order line recorded twice, typically through a double export, a retried upload, or a merge that ingested the same source twice. Exact duplicates are mechanically easy: pandas' duplicated method flags them, drop_duplicates removes all but the first occurrence, and the verification log entry is arithmetic — if inspection found forty-one duplicated rows, the cleaned file must have exactly forty-one fewer, and the removed revenue must equal the sum of the removed lines. The analytical stakes, however, are anything but minor, because duplicated order lines inflate revenue, order counts, and every per-customer metric downstream — and inflation is precisely the direction of StyleCraft's reconciliation gap in Section 4.1. Uniqueness, recall, is one of Section 3.7's six quality dimensions; this section is its repair shop.

A near-duplicate is the subtler form: two rows that record the same real-world event without being identical in every column. StyleCraft's extract contains the classic case, previewed in Table 3.4 — a block of orders re-exported from a source system with a new export timestamp, so that every substantive column matches but one administrative column differs. A check for fully identical rows sails past these untouched, which is exactly why they were planted. Detecting near-duplicates requires the analyst to decide which columns define the identity of an event: if two rows share order_id, order_line_id, customer_id, product_id, quantity, and price, they describe the same order line no matter what a load timestamp says, and duplicated with a subset argument restricted to the identity columns finds them. That decision — which columns constitute identity — is a judgment about the business, not about pandas, and it is the reason near-duplicate removal must be documented rather than merely executed: a subset chosen too narrowly deletes real repeat purchases (the same customer genuinely buying the same product twice in one day is rare but real), while a subset chosen too broadly leaves the inflation in place.

Duplication has a second and less obvious source, and it is worth naming here because it is the failure the labs guard against explicitly: duplicates the analyst creates. A join to a dimension table that is supposed to hold one row per customer will silently multiply transaction rows if the dimension in fact holds two, and the multiplication inflates revenue without raising an error or leaving an unmatched row to count. The remedy belongs to the same discipline as the rest of this section: state the expected cardinality of every join before running it, have the code enforce it, and check the row count on both sides. Section 4.11 shows the pattern, and Section 4.10 explains why AI-drafted merge code makes the check non-negotiable.

The verification log carries the burden of proof in every case. For each deduplication step: the number of rows flagged, the identity rule in one sentence, the predicted row count after removal, the actual count, and the change in total revenue — which, at the end of the pipeline, should account for a large share of the reconciliation gap the opening case discovered. If it does not, the log has just generated the chapter's most useful kind of finding: the gap has another cause, and the analyst knows to keep looking. With rows now unique, the pipeline turns to defects inside the values themselves: types, formats, and labels that disagree about how to spell the same fact.

4.7 Types, Formats, and Inconsistent Categories

Deduplication compares values, and comparison presumes the values are comparable. This section treats the defect family that breaks that presumption: values stored in the wrong type, formats that vary across source systems, and category labels that spell one fact several ways. Chapter 3 drew the conceptual line — stored data types are not measurement levels, and files routinely discard measurement information (Section 3.4.1) — and this section does the repairs that follow from it.

Type coercion is the conversion of a column from its stored type to the type its measurement requires: text to number, text to date, free text to categorical. The need arises because consolidated extracts inherit the habits of their sources, and operational systems export whatever their own storage held. StyleCraft's extract exhibits the canonical case: prices arriving as text with stray currency symbols — “$45.00” where 45.00 was meant — which a naive sum would refuse to add or, worse in other tools, silently drop. The repair pattern is normalize, then strip, then convert, then verify: put the column into a known text type first, because a consolidated extract frequently contains a mixture of strings and already-numeric values and a bare string accessor fails or misbehaves on the mixture; remove the non-numeric characters; convert with pd.to_numeric; and inspect what refused to convert. The final clause is where discipline lives. Conversion functions offer an errors=“coerce” convenience that turns unconvertible values into missing values, and used blindly it is a silent-data-loss machine: every stubborn string becomes a NaN without a word of complaint (pandas development team, 2026d). The verified pattern predicts how many values should fail conversion (from inspection), coerces, counts the new missing values, and requires the two numbers to match — a verification-log entry like any other.

Coercion has a companion problem that the same discipline solves: a numeric column that must also carry missing values. Setting a sentinel quantity to missing in an ordinary integer column silently converts the whole column to floating point, so that counts of units begin displaying as 1.0 and 2.0 and a column that measures a count stops looking like one. Pandas provides a nullable integer type for exactly this situation, and declaring it before introducing the missing value preserves the measurement meaning Chapter 3 insisted on (pandas development team, 2026b). The point is not cosmetic. A column's type is a claim about what kind of quantity it holds, and a repair that quietly changes the claim is a defect introduced by the cleaning.

Dates deserve the same care doubled, because they fail in more ways: multiple formats across source systems, day/month ambiguity, and values that parse cleanly into impossible facts. Chapter 3's conformity examples — an order_date before the customer's signup_date, or before the store's opening_date — parse without complaint; only a cross-table check against customers and stores exposes them, and the lab performs exactly that check.

Categorical standardization is the repair of labels that disagree about the same fact. StyleCraft's channel column contains “Online,” “online,” and “web”; its category labels include both “Occasionwear” and “occasion wear.” Chapter 3 named this a consistency defect and warned what a naive group-by does with it: three channels reported where one exists, with revenue split among them — a defect that does not change totals but silently corrupts every comparison, share, and chart built on the affected column. The repair is a mapping: an explicit, written dictionary from every observed variant to its canonical label, applied in code, with the log recording the value counts before and after and confirming that the total row count never moved. The mapping's authority is, once again, the data dictionary's allowed-values field: the canonical labels are not the analyst's invention but the documented vocabulary of the column. Two practices keep standardization honest. Normalize case and whitespace first, so the mapping enumerates real variants rather than typographical ones; and never map a variant whose meaning is uncertain — “web” probably means “Online,” but if any chance exists that it marks a distinct legacy channel, that question goes to the source-system owner, not to a guess.

DEFINITION

Type Coercion

Type coercion is the conversion of a column from its stored data type to the type required by its intended measurement level — for example, currency strings to numeric values, or date-like text to true dates — performed with an explicit accounting of every value that fails to convert. Coercion that silently discards or nullifies unconvertible values, or that changes what kind of quantity a column claims to hold, is a data defect of its own.

Source: Adapted from McKinney (2022) and Hellerstein (2008).

With rows unique, values typed, labels canonical, gaps treated, and extremes adjudicated, the file is clean in the sense this chapter can defend: every difference from the raw file is a logged, explained decision. The pipeline's

remaining work is constructive rather than corrective, and the next section turns to it — building the new columns and new grains the analyses ahead will need.

4.8 Derived Variables and Deliberate Grain Changes

Cleaning removes and repairs; preparation also builds. This section covers the two constructive operations that end most preparation pipelines: deriving new variables at the existing grain, and aggregating to a coarser grain on purpose. Both are routine pandas; both change what the table can claim; and the second, done carelessly, is the most consequential silent error in analytics.

A derived variable is a new column computed by an explicit rule from existing columns, at the grain of the table in which it appears. StyleCraft's schema already contains two — line_revenue is defined as quantity × unit_price × (1 − discount_pct), and line_cost as quantity × unit_cost — and Chapter 3's dictionary entry for line_revenue gave the standing advice this section repeats: recompute derived values rather than trusting stored ones. In a freshly cleaned file the recomputation is also a verification: computing line_revenue from its now-coerced, now-plausible inputs and comparing against the stored column confirms, row by row, that the type repairs and outlier treatments left the file arithmetically coherent. Beyond recomputation, preparation typically derives the small workhorse columns later chapters lean on: an order month extracted from order_date; a full-price flag from discount_pct equal to zero; a line-margin column from line_revenue minus line_cost. Each derivation is one line of code and one dictionary entry — a derived column without a recorded rule is tomorrow's mystery column — and each is logged with the simplest of checks, that the new column contains no unexplained missing values and the row count is unchanged.

Grouped aggregation changes the grain deliberately: it takes a table at one grain and produces a table at a coarser one, using pandas' groupby to collect rows sharing a key and summarize each group. The mechanics are compact — group transactions by order_id and sum line_revenue, and the result is an order-grain table with one revenue per order; group by customer_id and the result is customer-grain; group by store and month and the result feeds the time series of Chapter 10. The discipline is everything Chapter 3's Section 3.3 promised it would be. A grain change changes the claim: average line_revenue at line grain answers a different question than average order value at order grain, and computing “average order value” on the wrong grain is not a rounding disagreement but a different number with a different meaning. Every groupby therefore begins by naming its output grain — one row will equal one what — and choosing the summary each column deserves at that grain: quantities sum, dates take the first or the last, categorical attributes must be constant within the group or must not be carried along at all.

Aggregation gets its own verification pattern, and it is the most satisfying entry in the log because it is exact: totals that should survive the grain change must survive it. The sum of order-level revenue must equal the sum of line-level revenue to the cent; the count of distinct order_id values in the fact table must equal the row count of the order-grain table; the number of customers in a customer-grain roll-up must equal the number of distinct customers in the lines. When a predicted-equal total comes back unequal, the usual culprits are a key column with missing values — pandas excludes missing group keys by default, so groups vanish without comment unless the behavior is made explicit (pandas development team, 2026a) — or rows excluded upstream without a log entry. Aggregation also has a quieter trap that no total will reveal: a sum over a group whose values are all missing returns zero rather than missing, so an order whose only line has an unresolved quantity arrives at order grain reporting revenue of $0.00. The total still reconciles, and a genuinely unknown quantity has been converted into a confident zero. Section 4.11 shows how to keep the unresolved state visible. Either way, the mismatch, or its suspicious absence, is the log doing its job: it has caught, before any analysis, a discrepancy that would otherwise surface in a meeting.

DEFINITION

Derived Variable

A derived variable is a column computed from existing columns by an explicit, recorded rule, at the grain of the table in which it appears. Some derived variables are computed directly from columns at the current grain — line_revenue from quantity, unit_price, and discount_pct. Others exist only after an explicit aggregation has produced a new table at a coarser grain, as a customer's order count does in a customer-grain roll-up of transactions. A derived variable is complete only when its rule and its grain are documented in the data dictionary and its computation is verified against the totals it should preserve.

Source: Adapted from Kimball and Ross (2013) and Wickham (2014).

The file is now not only clean but productive: it carries the derived columns and companion grains the coming analyses need, and every one of them is documented. The pipeline's corrective and constructive work is done. What remains — before any formal method is chosen — is to look at what the data actually shows, and the next section takes up that discipline.

4.9 Exploratory Data Analysis: Disciplined Questioning

Preparation ends with a table the analyst can defend; analysis should not begin until the analyst has met it. This section introduces exploratory data analysis (EDA), the practice of systematically interrogating a fresh dataset — its distributions, its groups, its relationships — before committing to any formal method. The stance comes from Tukey, who argued that data analysis had overinvested in confirming hypotheses and underinvested in the prior work of finding out what the data suggests: exploratory work, in his phrase, is “detective work” (Tukey, 1977). The detective framing is exact. Exploration is not aimless looking; it is the disciplined pursuit of questions — What does this variable's distribution look like? Is that what the business story predicts? What is that bump? Who are those points? — in which every chart is produced to answer a stated question and every surprise is either explained or written down as a finding (Behrens, 1997).

EDA belongs in this chapter, adjoining preparation, for two reasons that practice keeps confirming. First, exploration is the final stage of verification: a distribution that looks wrong is often a defect the cleaning missed, and the fastest way to discover that a deduplication rule failed is to see the ghost of it in a histogram. Second, exploration is the first stage of every later chapter: the descriptive comparisons of Chapter 5, the segments of Chapter 6, and the models of Part II all begin from facts about shape and spread that exploration establishes. The habit this guide asks for is modest and permanent: no formal analysis on a table the analyst has not explored.

DEFINITION

Exploratory Data Analysis

Exploratory data analysis (EDA) is the systematic examination of a dataset's distributions, groups, and relationships — through summaries and simple charts — undertaken before formal modeling, in order to discover structure, surface anomalies, and generate the questions that later analysis will answer.

Source: Adapted from Tukey (1977) and Behrens (1997).

4.9.1 Distribution Shape: Skewness and Modality

The first exploratory question for any numeric variable is the shape of its distribution, and two aspects of shape do most of the descriptive work. Skewness is asymmetry: a right-skewed distribution has most of its values at the low end and a long tail of large values stretching right, and a left-skewed distribution mirrors that. Marketing quantities

are chronically right-skewed — order values, customer spend, campaign clicks all pile up small and tail off large — and StyleCraft's order revenue is right-skewed by design: most baskets are one or two items in a catalog that tops out near $90, while suburban occasionwear baskets stretch the tail. Recognizing skew matters immediately for verification (a symmetric order-value distribution in this data would itself be an anomaly worth chasing) and matters shortly for description, because summarizing skewed data raises choices Chapter 5 takes up. Modality is the number of humps: a unimodal distribution has one concentration of values, a bimodal distribution two. Bimodality is one of exploration's most valuable alarms, because it usually means the variable is describing a mixture of populations rather than one — and in StyleCraft's data, where two customer populations were designed in, a two-humped spending distribution is not noise but the company's central strategic fact showing itself in a histogram.

4.9.2 Reading Exploratory Charts for Anomalies

Exploration works through four chart types, and this guide asks them to be read, at this stage, in a particular way: for anomalies and questions, not for polish. The theory of chart selection and visual encoding — why these forms work and how to design them for an audience — is Chapter 12's territory; here the charts are instruments, and the skill is knowing what each instrument detects.

A histogram shows one numeric variable's distribution by binning values and drawing a bar per bin. Read it for shape (skew, modality, per Section 4.9.1), for spikes (a bar taller than its neighbors at a suspiciously round value — a price exactly at a sentinel, a quantity exactly at a cap), and for gaps and orphans (empty ranges, or a lone bar far from the body, which is an outlier rule made visible).

A box plot draws the median and the quartile box, and it is Section 4.5's IQR rule made visible (Tukey, 1977). Under the common Tukey convention that plotting libraries implement by default, the whiskers do not sit at the fences themselves: they extend to the most extreme observed values that still lie within 1.5 interquartile ranges of the first and third quartiles, and observations beyond them are drawn individually as possible outliers (Matplotlib Development Team, 2026). The distinction matters when reading a picture as evidence, because the whisker tip is a real observation in the data rather than a computed boundary. The box plot's particular strength is comparison: box plots of order value by store type put the suburban-urban contrast, spread and outliers included, in one glance. Read it for medians that differ across groups, boxes of very different heights (heterogeneity across groups), and dense swarms of flagged points (a tail, or a defect).

A bar chart compares a summary statistic across categories; at the exploratory stage its most important service is humble — a bar per category label is the fastest way to see that “Online,” “online,” and “web” survived the cleaning, or that one store's revenue is implausibly zero. Read it for categories that should not exist, categories missing that should exist, and magnitudes wildly out of family. A scatterplot shows the joint behavior of two numeric variables, one point per row. Read it for the expected structure (line_revenue rising with quantity along plausible price bands), for points off the structure (a high revenue at quantity one beyond any catalog price — an arithmetic defect surviving), and for clumps that suggest subpopulations. In every case the reading ends the same way: each anomaly becomes a question, each question gets pursued to an explanation — a defect, a documented business fact, or a genuine discovery — and the discoveries are written down, because an insight that lives only in a notebook scroll is not yet evidence.

Exploration, so practiced, closes the preparation story and opens the analysis story. Before the labs perform both on StyleCraft's files, one more section addresses the collaborator that will write most of the code: the AI assistant.

4.10 AI as a Data Preparation Assistant

The pipeline of Sections 4.2 through 4.9 is, line for line, the kind of work AI assistants perform fluently: loading files, coercing types, mapping labels, deduplicating, grouping, plotting. Consistent with the guide's stance (Sections 1.9, 2.8, and 3.9), this section identifies where that fluency genuinely helps, where delegation must stop, and how this chapter's verification theme — the log — makes the collaboration safe.

The genuine help is broad. An assistant given the data dictionary and a description of a defect writes serviceable cleaning code in seconds: the strip-and-coerce pattern for currency strings, the subset-keyed deduplication, the

groupby with the right aggregations. It drafts inspection summaries, proposes standardization mappings from observed value counts, generates the boilerplate of every exploratory chart, and — often underrated — explains unfamiliar code the analyst has inherited. For the mechanical majority of preparation, the assistant is a competent junior pair of hands, and this course expects students to use it as one.

The failure modes are equally specific, and three deserve names because the lab plants encounters with each. Silent row drops: cleaning code frequently discards rows as a side effect — errors="coerce" followed by a dropna, a filter written wider than intended, a groupby key containing missing values — and AI-generated code is prone to including such steps helpfully and without comment, so that the file shrinks and nothing announces it. Wrong join keys: asked to check order dates against signup dates, an assistant must join transactions to customers, and a join on the wrong key, the wrong key type (a customer_id string in one table, a stray-whitespace variant in the other), or a dimension table that is not in fact unique on the key produces a result that runs, returns rows, and is quietly wrong — the most dangerous kind of wrong, because nothing errors and, in the duplicated-key case, the row count moves in the direction that flatters revenue. Unstated imputation defaults: asked to "handle" missing values, assistants default to a remedy — usually mean or modal filling — without surfacing that a choice among Section 4.4's remedies was made at all, importing the corresponding bias unannounced. The common thread is the one this guide keeps finding: the assistant optimizes for code that runs and output that looks complete, while the analyst is accountable for what the code did to the data — and in preparation, what the code did to the data is the entire question.

The countermeasure is the verification log, applied to delegated code exactly as to handwritten code — which is to say, the log is not extra work occasioned by AI; it is the same discipline, now doing double duty as an audit. Before running any AI-generated cleaning step, the analyst writes the prediction: what the step should change, what it should not, and by how much. After running it, the actuals either match or the code goes back. Row counts before and after every delegated step are non-negotiable; a join is always accompanied by a stated expected cardinality, an enforced check that the code makes the join keep it, and a count of matched and unmatched rows; an imputation is always preceded by the question "which remedy is this, and why?" Every exchange is documented per Appendix D.

> **AI IN PRACTICE**
>
> **Delegate the Code, Never the Count**
>
> A productive pattern for delegated cleaning: paste the relevant data dictionary entries and a sample of rows (never confidential data; the synthetic StyleCraft files are safe), then prompt — "Write pandas code to [one cleaning step]. Before the code, state exactly what the step will change: which columns, how many rows affected, what the row count and revenue total should be afterward, and what the code deliberately leaves alone. If the step is a join, state the expected cardinality and enforce it in the code. List any rows it would drop and why." Run the code only after recording the assistant's predictions and your own in the verification log; if the assistant cannot state what its code will do to the row count, that is the audit failing before the code runs. Compare actuals, investigate every mismatch, and document the exchange per Appendix D.

One boundary completes the section. The assistant may draft every line of cleaning code in this chapter; it may not make the editorial decisions the code executes. Whether "web" means "Online," whether the 999s are sentinels, whether in-store shoppers' missing ages are dropped, filled, or flagged, which columns define a duplicate — these are judgments about StyleCraft's business, they are exactly the decisions Section 4.14 will argue carry ethical weight, and they belong to the analyst of record. The labs that follow are structured around that division of labor.

4.11 Hands-On Application in Python and Google Colab

The preceding sections built the pipeline conceptually. This section runs it. Lab 4.1 works through the full preparation sequence twice: first on a fourteen-row miniature of transactions_raw constructed in the notebook, so that every count and total can be verified by hand, and then on StyleCraft's full transactions_raw file from the

companion repository, against the same defect families at scale. Lab 4.2 explores the cleaned file. Throughout, follow the course division of labor: AI assistants may draft the code (Appendix C has prompting templates; Appendix A covers Colab mechanics), but every step gets a verification-log entry — prediction first — and every AI exchange is documented per Appendix D.

4.11.1 Lab 4.1, Part A: The Dirty Dozen (and Two More)

The miniature below contains fourteen order lines and, planted among them, one instance of each defect family from the chapter: a currency-string price, two label variants in channel, an exact duplicate, a suspected re-export pair that also contains an error, a quantity sentinel, an out-of-band price, a missing discount, a missing campaign attribution, and an order date that precedes the customer's signup. Before running the cell, read the code and start the verification log: predict the shape, and predict which routine from Section 4.3 will surface which defect.

Note the cell's last two statements before the shape check. The raw frame is built once and then copied, and every step that follows operates on the copy, tx. This is the raw-stage obligation of Table 4.1 written as executable code: at the end of the pipeline, tx_raw still holds the currency strings, the label variants, and the 999, so any log entry can be re-checked against the evidence it claims to have repaired.

Code 4.1. Create the transactions_raw miniature and preserve it

```
import pandas as pd
import numpy as np

tx_raw = pd.DataFrame({
    "order_id":      ["O1001","O1001","O1002","O1003","O1004",
                      "O1004","O1005","O1006","O1006","O1007",
                      "O1008","O1009","O1009","O1010"],
    "order_line_id": ["L1","L2","L3","L4","L5","L6","L7","L8",
                      "L8","L9","L10","L11","L12","L13"],
    "customer_id":   ["C01","C01","C02","C03","C04","C04","C05",
                      "C06","C06","C07","C08","C09","C09","C10"],
    "product_id":    ["P010","P022","P031","P010","P044","P051",
                      "P022","P063","P063","P031","P044","P075",
                      "P075","P010"],
    "store_id":      ["ONLINE","ONLINE","S01","APP","S07","S07",
                      "ONLINE","S01","S01","ONLINE","S09","ONLINE",
                      "ONLINE","APP"],
    "channel":       ["Online","Online","Store","App","Store",
                      "Store","online","Store","Store","web",
                      "Store","Online","Online","App"],
    "order_date":    ["2025-11-03","2025-11-03","2025-11-04",
                      "2025-11-04","2025-11-05","2025-11-05",
                      "2025-11-06","2026-01-06","2026-01-06",
                      "2025-11-07","2025-11-08","2025-11-08",
                      "2025-11-09","2024-06-15"],
    "quantity":      [1, 2, 1, 1, 3, 2, 1, 1, 1, 2, 999, 1, 1, 1],
    "unit_price":    ["$38.00","$29.00","$52.00","$38.00","$74.00",
                      "$81.00","$29.00","$45.00","$45.00","$52.00",
                      "$36.00","$240.00","$68.00","$38.00"],
    "discount_pct":  [0.0, 0.20, 0.0, np.nan, 0.0, 0.0, 0.20,
                      0.0, 0.0, 0.30, 0.0, 0.0, 0.0, 0.0],
    "campaign_id":   ["CMP04","CMP04", None, "CMP11", None, None,
                      "CMP04", None, None, "CMP09", None, None,
                      None, "CMP11"]
})

customers_mini = pd.DataFrame({
    "customer_id": ["C01","C02","C03","C04","C05",
                    "C06","C07","C08","C09","C10"],
    "signup_date": pd.to_datetime([
```

```
        "2024-05-12","2024-10-01","2025-09-20","2025-07-30",
        "2024-08-15","2025-12-28","2025-10-05","2024-07-01",
        "2025-06-22","2025-01-10"])
})

# The raw object is evidence. Never transform it in place;
# every preparation step below runs on the working copy, tx.
tx = tx_raw.copy(deep=True)

tx_raw.shape
```

Expected output: (14, 11) — fourteen order lines, eleven columns. Log entry one: baseline rows = 14.

Input: two literal frames, the fourteen-line extract and the ten-customer dimension table reused from Lab 3.1, so the signup dates are already familiar. Transformation: none yet, apart from the deep copy that separates the evidence from the workspace. Output: the baseline the whole log is checked against. Every later count in this lab is a claim about how far the file has moved from (14, 11), and tx_raw is what makes that claim auditable rather than remembered.

Now run the inspection routines of Section 4.3 in order — dtypes, head, info, describe — predicting each output before executing.

Code 4.2. Inspect before touching

```
# A notebook renders only a cell's last expression, so each
# inspection result is displayed explicitly.
from IPython.display import display

display(tx.dtypes)
display(tx.head(14))
tx.info()
display(tx.select_dtypes(include="number").describe())
```

Expected output, worth predicting in detail: unit_price reports object, not a number — the currency strings have made the entire column text, so the numeric-only describe excludes it entirely, an absence that is itself a finding. quantity reports a maximum of 999. discount_pct reports 13 non-null values of 14. The head reveals the channel variants and the repeated L8 line.

Input: the untouched working copy. Transformation: none — inspection is a read-only stage, which is why it can be run before any prediction has been risked. Output: four diagnostic views, each displayed explicitly. A notebook automatically renders only the last expression in a cell, so a cell listing four routines shows one of them; display() is used here, and wherever a later cell reports more than one result, so that the outputs the prose asks students to read are the outputs students actually see. Taken together the four produce the written diagnosis. For the miniature it reads: one type defect (unit_price), one label defect (channel), at least one exact duplicate (the repeated L8 row), a quantity sentinel, one missing discount_pct, and — pending checks — possible near-duplicates, an out-of-band price hiding inside the text column, and date inconsistencies. The pipeline now runs in the default order of Section 4.2: types and labels, duplicates, impossible values, missing values, derivations.

Code 4.3. Step 1 — coerce unit_price from currency strings

```
before_na = tx["unit_price"].isna().sum()

# Normalize to text first: a real extract may mix strings with
# values that already arrived numeric.
price_text = tx["unit_price"].astype("string")

clean_price = price_text.str.replace(r"[$,]", "", regex=True)

tx["unit_price"] = pd.to_numeric(clean_price, errors="coerce")
```

```
after_na = tx["unit_price"].isna().sum()
print(before_na, "→", after_na, "| rows:", len(tx))
print("max price:", tx["unit_price"].max())
```

Expected output: 0 → 0 | rows: 14, and a maximum price of 240.0.

Input: a text column holding fourteen currency strings. Transformation: normalize to the string dtype, strip the dollar signs and any thousands separators, then convert with errors="coerce" so that failures become missing values the code can count rather than exceptions that stop the notebook. Output: a numeric column, an unchanged row count, and a maximum that now announces the out-of-band price the object dtype was hiding. Confirm that tx_raw["unit_price"] still reads "$38.00" in its first row: the repair happened on the copy.

> **VERIFICATION CHECK**
>
> Before you run: every one of the fourteen strings is a dollar sign followed by a clean number, so all fourteen should convert. Predict missing values in unit_price going from 0 to 0, a row count that stays at 14, and — reading the raw strings — a maximum of $240.00 once the column is numeric.
>
> After you run: before_na and after_na both report 0, len(tx) reports 14, and the maximum is 240.0.
>
> Investigate if: after_na comes back greater than zero. That means errors="coerce" has silently nulled a value the prediction said would convert; find the offending string in tx_raw before proceeding, because the raw column is still intact.

Code 4.4. Step 2 — standardize the channel labels

```
print("Before standardization")
display(tx["channel"].value_counts(dropna=False))

channel_map = {
    "Online": "Online",
    "online": "Online",
    "web": "Online",
    "Store": "Store",
    "App": "App",
}
tx["channel"] = tx["channel"].map(channel_map)

print("After standardization")
display(tx["channel"].value_counts(dropna=False))
```

Expected output: before — five labels (Store 6, Online 4, App 2, online 1, web 1); after — three labels (Online 6, Store 6, App 2). Row count unchanged at 14, and no missing values in channel.

Input: a categorical column carrying five spellings of three channels. Transformation: an explicit mapping from every observed variant to its canonical label, built from the value counts rather than from memory. Output: two labeled tallies, before and after, because the comparison between them is the verification — three canonical labels, the same fourteen rows, and no new missing values. A missing value after mapping would mean the enumeration was incomplete — map returns NaN for any key it was not given — which is precisely why the value counts are read before the map is written. The mapping decision itself, that "web" is the Online channel and not a distinct legacy channel, is recorded in the log as a judgment call, with its source (in the course setting, the data dictionary; at StyleCraft, the e-commerce platform owner).

Code 4.5. Step 3 — remove exact duplicates, then near-duplicates

```
exact = tx.duplicated().sum()
print("exact duplicates:", exact)
```

```
tx = tx.drop_duplicates().copy()
print("rows:", len(tx))

identity = ["order_id","customer_id","product_id",
            "quantity","unit_price"]
near = tx.duplicated(subset=identity).sum()
print("near-duplicates on identity columns:", near)
tx = tx.drop_duplicates(subset=identity).copy()
print("rows:", len(tx))
```

Expected output: exact duplicates: 1, rows: 13; near-duplicates: 0, rows: 13.

Input: fourteen rows with canonical labels and numeric prices — the order matters, because a duplicate check run before Step 1 would have compared "$45.00" to "$45.00" as text and a check run before Step 2 would have treated "online" and "Online" as different events. Transformation: drop rows identical in every column, then drop rows identical on the declared identity columns. Output: thirteen rows and a near-duplicate count of zero — which is the correct result for this rule and a deliberately uncomfortable one, as the Verification Check explains.

VERIFICATION CHECK

Before you run: two predictions. First, exactly one row is a full copy of another (the repeated L8 line), so exact = 1 and the row count falls 14 → 13. Second, order O1009 appears twice with the same customer and product but different order_line_id, order_date, and — check the raw strings — different prices ($240.00 and $68.00). Ask whether the identity rule above catches it, and write the answer down before running.

After you run: exact = 1, rows 13; near = 0, rows still 13. The identity rule does not catch the O1009 pair, because unit_price is in the identity list and the prices differ.

Investigate if: near comes back greater than zero, or the row count falls below 13. Either means the identity columns are catching events the rule was not meant to catch. Record the lesson either way: near-duplicate detection and error correction interact, and the O1009 pair is resolved by Step 4's domain-limit check instead. Exercise 4.7 revisits this trap.

Code 4.6. Step 4 — impossible values, a validated join, and dates

```
# Domain limits from the data dictionary: prices $24-$90,
# plausible basket quantities.
bad_price = tx[(tx["unit_price"] < 24) | (tx["unit_price"] > 90)]
bad_qty   = tx[tx["quantity"] > 20]
print(bad_price[["order_id","order_line_id","unit_price"]])
print(bad_qty[["order_id","order_line_id","quantity"]])

# Cross-table date check. State the expected cardinality and
# make the code enforce it: one customer row per transaction.
tx["order_date"] = pd.to_datetime(tx["order_date"])
rows_before_join = len(tx)

tx = tx.merge(
    customers_mini,
    on="customer_id",
    how="left",
    validate="many_to_one",
    indicator=True
)

assert len(tx) == rows_before_join, "the join changed the row count"
print(tx["_merge"].value_counts())
```

```
print("unmatched customers:", tx["signup_date"].isna().sum())
tx = tx.drop(columns="_merge")

bad_dates = tx[tx["order_date"] < tx["signup_date"]]
print(bad_dates[["order_id","customer_id",
                 "order_date","signup_date"]])
```

Expected output: one out-of-band price (O1009's $240.00 line, L11), one sentinel quantity (the 999 on O1008, L10), a merge indicator reporting 13 both and 0 left_only, zero unmatched customers, and one impossible date — O1010, ordered 2024-06-15 by C10, who signed up 2025-01-10.

Input: the deduplicated thirteen-row frame and the customer dimension. Transformation: three diagnostic filters and one join. The join is where this cell earns its keep. Counting unmatched rows after a merge, as an earlier draft of this lab did, detects a customer who is missing from the dimension table; it does not detect a customer who appears in it twice. A duplicated key silently multiplies transaction rows, inflating revenue while producing no error and no unmatched row to count — the wrong-join-key failure of Section 4.10 in its most flattering disguise. Declaring validate="many_to_one" makes pandas raise if the right-hand key is not unique, and the assertion on the row count states the same expectation in the log's own vocabulary. Output: the flagged rows, an explicit matched/unmatched tally from the indicator column, and a row count that has provably not moved. The indicator column is dropped once it has been read, because a verification artifact does not belong in the certified table.

Each flagged row now gets classified per Section 4.5, and the classifications are the log's judgment entries. The $240.00 line, sitting beside its $68.00 twin on the same order, is resolved as the erroneous member of a suspected re-export pair. Note what the file alone can and cannot establish here. The domain limit establishes that $240.00 is out of band and therefore wrong; it does not establish what the right value was, and $240.00 is not a decimal shift of $68.00, so the two lines cannot be reconciled by arithmetic. The defect catalog — or, at StyleCraft, the source-system record — identifies L12 as the valid line, which is what licenses the judgment: exclude L11 and retain the plausible line. Where no such external record exists, the defensible move is not to guess which line is right but to set the out-of-band value to missing and say so. The 999 quantity has no corroborating basket — it is treated as a sentinel and set to missing, converting it into Section 4.4's problem with a known mechanism. The impossible date cannot be repaired from the file alone; the ground rule is documented exclusion, so the O1010 line is removed under a written rule. Note what the miniature keeps: the big-but-legitimate basket on O1004 (three units at $74 plus two at $81) survives every rule, because nothing about it is an error — it is the occasionwear tail, and deleting it would have deleted the phenomenon. The next cell applies the three classifications; predict the row counts before running it.

Code 4.7. Step 4b — apply the classifications

```
# Judgment 1: the defect catalog identifies L11 as the
# erroneous member of the O1009 re-export pair.
tx = tx[tx["order_line_id"] != "L11"].copy()
print("rows after removing the erroneous re-export line:", len(tx))

# Judgment 2: the 999 quantity is a sentinel. Declare the
# nullable integer type first, so a count stays a count.
tx["quantity"] = tx["quantity"].astype("Int64")
tx.loc[tx["quantity"] > 20, "quantity"] = pd.NA
print("quantity dtype:", tx["quantity"].dtype,
      "| missing quantities:", tx["quantity"].isna().sum())

# Judgment 3: documented exclusion of the impossible-date line.
tx = tx[~(tx["order_date"] < tx["signup_date"])].copy()
print("rows after the date exclusion:", len(tx))
```

Expected output: rows fall 13 → 12 after the re-export removal and 12 → 11 after the date exclusion; quantity reports dtype Int64 with one missing value. Setting the sentinel to missing changes no row count, only the missing-value tally. Three rows of the log now carry judgment flags.

Input: thirteen rows carrying three classified defects. Transformation: one row removed under an error rule, one value set to missing under a sentinel rule, one row removed under an exclusion rule. Output: eleven rows, each departure explained. The type declaration deserves a sentence of its own. Assigning a missing value into an ordinary integer column promotes the column to floating point, and quantities then display as 1.0 and 2.0 — a count that no longer looks like a count. Pandas' nullable Int64 type exists for integer data that may contain missing values (pandas development team, 2026b), and declaring it before the assignment preserves what the column measures. This is the Section 4.7 principle applied to the cleaning's own side effects: a repair that changes a column's claim about its quantity is a defect the repair introduced.

Code 4.8. Step 5 — treat the remaining missing values and derive

```
tx["quantity_missing"] = tx["quantity"].isna()
tx["discount_missing"] = tx["discount_pct"].isna()
tx["discount_pct"] = tx["discount_pct"].fillna(0.0)

tx["line_revenue"] = (tx["quantity"] * tx["unit_price"]
                      * (1 - tx["discount_pct"]))

print("rows:", len(tx))
print("computable lines:", tx["line_revenue"].notna().sum())
print("known revenue:", round(tx["line_revenue"].sum(), 2))
```

Expected output: rows: 11, computable lines: 10, known revenue: 767.4.

Input: eleven surviving rows, one with an unresolved quantity and one that had a missing discount. Transformation: two flag columns, one justified fill, and the recomputation of line_revenue from its repaired inputs. Output: eleven rows and a revenue figure that is not the revenue of eleven rows. Pandas' sum excludes missing values by default (pandas development team, 2026c), so $767.40 is the total of the ten computable lines; the line whose quantity became missing in Step 4b contributes nothing and is not counted as zero either. Every student should reproduce $767.40 with a calculator from the ten surviving computable lines.

Two decisions in this cell are judgment entries rather than arithmetic. Filling a missing discount_pct with 0 asserts that the line was sold at full price, which is a business judgment — the modal case, per the dictionary — and not a statistical necessity; the flag column preserves the ability to test its sensitivity later. The missing campaign_id values are left exactly as they are: Section 4.4's two-kinds-of-null problem cannot be resolved from inside the file, the dictionary records the ambiguity, and imputing an attribution would manufacture marketing performance.
One step remains, and it is the step that separates a cleaned file from a certified one. Eleven rows survive, every departure from the raw file is explained, and the arithmetic is correct — and yet one surviving row carries a revenue value that is not merely unknown but undecided, because the sentinel quantity was converted to missing rather than resolved. Section 4.2.1 defined certification as a threshold: a file is certified for a stated use when no quantity that use depends on is left unresolved. Revenue analysis depends on line_revenue. The gate below makes the threshold executable rather than rhetorical.

Code 4.9. Step 6 — the certification gate

```
unresolved = int(tx["line_revenue"].isna().sum())
known_revenue = tx["line_revenue"].sum()

print("surviving rows:", len(tx))
print("rows with unresolved revenue:", unresolved)
print("known revenue across computable lines:",
      round(known_revenue, 2))

if unresolved == 0:
    print("STATUS: certified for revenue analysis.")
else:
```

```
    print("STATUS: provisionally cleaned —", unresolved,
          "row(s) carry unresolved revenue.")
```

Expected output: surviving rows: 11, rows with unresolved revenue: 1, known revenue across computable lines: 767.4, and the status line reporting provisionally cleaned.

Input: the prepared eleven-row table. Transformation: none — the gate changes nothing, which is the point of a gate. Output: a verdict. Part A does not certify. It closes as provisionally cleaned, with one unresolved quantity, and reports known revenue across ten computable lines of $767.40. The distinction is not pedantry. A table offered as certified is a table whose totals a CFO may use without asking further questions, and $767.40 is not the miniature's revenue; it is the revenue of the part of the miniature that could be computed. Resolving the eleventh line requires something the file does not contain — the true quantity from the source system, or a documented decision to exclude the line — and until one of those arrives, the honest label is the provisional one. Close Part A's log accordingly: 14 rows in, 11 out, every departure explained, one quantity outstanding, and revenue computable for ten of eleven surviving lines.

In Part B, where the full file must support the expansion review, the same gate is written as an assertion rather than a status line, because the answer key admits no unresolved revenue: assert transactions_clean[“line_revenue”].isna().sum() == 0, with a message naming certification as the thing being blocked. A pipeline that cannot pass that line has not finished, however clean its output looks.

4.11.2 Lab 4.1, Part B: The Full File

Part B applies the same pipeline to StyleCraft's actual transactions_raw file, available with customers, products, and stores from Brightspace and the companion repository. The defects are the same families at realistic scale, and the file ships with two anchors the miniature could not offer: the transactions_clean answer key, against which the final table is compared, and the known totals (row counts, distinct orders, total revenue) that transactions_clean implies.

Work through the pipeline in the same order, with four additions the scale demands. First, copy the raw frame before the first transformation, exactly as Code 4.1 does, and keep it in memory to the end. Second, validate every join. Part B joins transactions to customers, to products, and to stores, and each of the three is a many-to-one relationship that the code should declare and enforce with the validate and indicator arguments of Code 4.6, bracketed by a row count before and an assertion after. Extend the impossible-date check to stores by joining on store_id and flagging order dates before the store's opening_date, remembering that ONLINE and APP are reserved store_id values with no opening date — predict how the join treats them before running it. Third, let an AI assistant draft each step's code per the AI in Practice pattern, and keep its predictions and yours side by side in the log; at full scale, at least one delegated step will disagree with your prediction, and finding out why is the assignment. Fourth, close the log with the certification entries: the cleaned file's row count, distinct-order count, and total revenue against transactions_clean's; the certification gate of Code 4.9 written as an assertion; and the reconciliation that opened the chapter — how much of the raw file's excess revenue the removed duplicates and corrected errors account for.

> **VERIFICATION CHECK**
>
> Before you run: predict, for each of the three dimension joins, the row count afterward (unchanged), the number of unmatched rows, and which reserved store_id values you expect to appear as unmatched by design. Predict the final row count, distinct-order count, and total revenue before comparing them to transactions_clean.
>
> After you run: every join leaves the transaction row count unchanged; the unmatched counts match your predictions or are explained; the certification assertion on unresolved revenue passes; and the final totals reconcile with the answer key.

> Investigate if: any difference from the key is not attributable to a logged decision. The comparison against transactions_clean is graded self-verification, in keeping with the book's known-truth verification thread: the answer key exists so the pipeline can be checked, not copied. Matching totals do not require matching code — two defensible pipelines can differ in judgment entries — but an unexplained difference of even one row means a silent step ran somewhere. Find it.

4.11.3 Lab 4.2: First Exploration of the Cleaned File

Lab 4.2 explores transactions_clean (or your certified equivalent), practicing Section 4.9's questioning discipline. Charts here use the basic matplotlib patterns from Appendix A; design is not the point, questions are — the theory of chart selection and encoding is Chapter 12's. Four worked cells follow, each beginning with a written prediction and ending with the anomalies the picture raises. Run them on the full cleaned file; the miniature is too small to have a distribution worth reading.

The lab opens with a deliberate grain change, because every chart after it is drawn at order grain rather than line grain and the difference is the whole of Section 4.8. The aggregation below also carries two verifications and one deliberate choice worth reading closely before running the cell.

Code 4.10. Aggregate order lines to order grain

```
# Precondition 1: this cell claims a certified input.
assert transactions_clean["line_revenue"].isna().sum() == 0, (
    "order aggregation requires a table certified "
    "for revenue analysis"
)

# Precondition 2: "first" is only honest if the attribute is
# constant within the order. Check, do not assume.
for column in ["store_type", "channel"]:
    constant = (
        transactions_clean
        .groupby("order_id", dropna=False)[column]
        .nunique(dropna=False)
        .le(1)
        .all()
    )
    assert constant, f"an order carries multiple {column} values"

# One row will equal one order.
order_summary = (
    transactions_clean
    .groupby("order_id", as_index=False, dropna=False)
    .agg(
        order_revenue=("line_revenue",
                       lambda s: s.sum(min_count=1)),
        lines=("line_revenue", "size"),
        unresolved_lines=("line_revenue",
                          lambda s: s.isna().sum()),
        store_type=("store_type", "first"),
        channel=("channel", "first"),
    )
)

# Verification 1: one row per distinct order.
assert len(order_summary) == \
    transactions_clean["order_id"].nunique(dropna=False)

# Verification 2: the grain change preserves the total.
```

```
assert np.isclose(
    order_summary["order_revenue"].sum(),
    transactions_clean["line_revenue"].sum()
)

print("orders:", len(order_summary))
print("order revenue total:",
      round(order_summary["order_revenue"].sum(), 2))
print("orders with unresolved lines:",
      int((order_summary["unresolved_lines"] > 0).sum()))
```

Expected output: an order count equal to the distinct order_id count, an order-grain revenue total equal to the line-grain total to the cent, and zero orders carrying unresolved lines. All four assertions pass silently.

Input: the certified line-grain table. Transformation: rows are collected by order_id and summarized, with each column given the summary its meaning deserves — revenue sums, lines count, the dimension attributes take the first value within the order. Output: an order-grain table and four proofs that the grain change was faithful. The assertions are the cell, and each states an expectation the prose would otherwise leave as an assumption. The first makes the input requirement executable: this cell says it consumes a certified table, and a certified table has no unresolved revenue, so the claim is checked rather than trusted. The second guards the word first. Taking the first store_type in a group is honest only if every line in the order carries the same one; if a contaminated order held two, the aggregation would silently keep whichever appeared first and the order would be filed under a store type it only partly belongs to. The remaining two assertions are the grain-change identities of Section 4.8: one row per distinct order, and a total preserved to the cent.

Two arguments in the aggregation deserve the same attention. The dropna=False argument makes the treatment of missing group keys explicit rather than default, because pandas otherwise drops them without comment (pandas development team, 2026a) and orders would disappear silently. The min_count=1 argument keeps an order whose lines are all unresolved as missing instead of collapsing it to $0.00 — the trap Section 4.8 named, in which an unknown quantity is quietly promoted to a confident zero that no total will catch, because zero adds nothing to a sum. On a certified input the first assertion makes that trap unreachable, and the pattern is kept anyway, for two reasons: it is the habit that protects the analyst on the day the input is provisional rather than certified — Part A's table is exactly such a case — and the unresolved_lines column carries the certification question up to the new grain, so that the order table can be gated the same way the line table was.

VERIFICATION CHECK

Before you run: predict the number of orders (the distinct order_id count from your certified file), predict that the order-grain revenue total will equal the line-grain total exactly, and state — before the code tells you — whether you expect any order to contain more than one store_type or channel, and why.

After you run: all four assertions pass silently, and the printed order count and revenue total match the predictions.

Investigate if: any assertion fails. A failed certification check means the input is provisional and the gate of Code 4.9 was never cleared. A failed constancy check means an order spans two store types or channels, which is a defect in the upstream join or a real business case the aggregation must handle deliberately rather than by taking whichever value came first. An order count above the distinct-order count usually means a duplicated key in an upstream join, and a revenue total below the line-grain total usually means a group key containing missing values. A total that matches while some order reports $0.00 revenue means min_count was omitted and unresolved lines were summed as zero.

With the grain settled, the first question is the shape of the order-value distribution. Write the prediction first: from the business story, order revenue should be right-skewed and bounded below, with a long tail produced by occasionwear baskets, and — because StyleCraft was designed with two customer populations — a second hump would not be a surprise so much as a confirmation.

Code 4.11. Histogram of order revenue

```
import matplotlib.pyplot as plt

revenue = order_summary["order_revenue"].dropna()

fig, ax = plt.subplots(figsize=(7, 4))
ax.hist(revenue, bins=40, edgecolor="white")
ax.set_xlabel("Order revenue ($)")
ax.set_ylabel("Number of orders")
ax.set_title("Distribution of order revenue")
plt.show()

print(revenue.describe().round(2))
print("orders dropped as unresolved:",
      len(order_summary) - len(revenue))
```

Expected output: a right-skewed histogram concentrated at the low end with a thinning tail to the right, a mean above the median in the describe output, and a count of dropped orders equal to the unresolved-order count from Code 4.10.

Input: the order-grain revenue column. Transformation: values are binned and counted; the dropna is stated explicitly and its cost is printed, because a chart that silently omits rows is the histogram version of a silent row drop. Output: a picture, a five-number summary to read alongside it, and an accounting of what the picture excludes. Read the picture in the order Section 4.9.2 prescribes: shape first (is the skew there, and does the mean sit above the median as skew implies?), then spikes (a bar standing above its neighbors at a suspiciously round value is a sentinel or a cap that survived cleaning), then gaps and orphans (an empty range, or a lone bar far to the right, is an outlier rule made visible). Each anomaly becomes a log entry with a resolution: a defect the cleaning missed, a documented business fact, or a finding. A second hump belongs in the third category — it is the two designed customer populations showing themselves, and Chapter 6 will name them.

The histogram describes the file as a whole. The next question is comparative, and it is the one the expansion review actually asks: do suburban stores generate larger orders than urban stores, and is the difference in the middle of the distribution or in its tail? Predict the answer before drawing it — which group has the higher median, and which has the longer tail — because a prediction written down is what turns the picture into evidence rather than decoration.

Code 4.12. Grouped box plots of order revenue

```
def revenue_by(frame, column):
    """Return (labels, arrays) for a box plot grouped by column."""
    labels, groups = [], []
    for key, part in frame.groupby(column, dropna=False):
        labels.append(str(key))
        groups.append(part["order_revenue"].dropna().to_numpy())
    return labels, groups

for column, title in [("store_type", "store type"),
                      ("channel", "channel")]:
    labels, groups = revenue_by(order_summary, column)

    fig, ax = plt.subplots(figsize=(7, 4))
    ax.boxplot(groups, whis=1.5)
    ax.set_xticks(range(1, len(labels) + 1))
    ax.set_xticklabels(labels)
```

```
    ax.set_ylabel("Order revenue ($)")
    ax.set_title("Order revenue by " + title)
    plt.show()

    print(order_summary.groupby(column, dropna=False)
          ["order_revenue"]
          .agg(["count", "median", "mean", "max"]).round(2))
```

Expected output: two box plots, one per grouping, each accompanied by a table of group counts, medians, means, and maxima. The suburban group should show both a higher median and a longer upper tail than the urban group; every group's mean should sit above its median.

Input: the order-grain table and two grouping columns. Transformation: orders are split into groups, each group's quartiles are computed, and the whiskers are drawn under the default Tukey convention — extending to the most extreme observations lying within 1.5 interquartile ranges of the quartiles, with anything beyond drawn as an individual point (Matplotlib Development Team, 2026). Output: a comparison that a pair of averages would flatten, plus the numbers behind it. Read the boxes for medians that differ, for boxes of very different heights (one group is more heterogeneous than another), and for dense swarms of plotted points above the upper whisker. That last pattern is the one this chapter has been arguing about: those points are the occasionwear baskets, they are legitimate extremes rather than defects, and an analyst who had trimmed them in Section 4.5 would now be looking at a picture with the expansion strategy's best evidence removed from it.

The final chart returns to line grain and to arithmetic. A scatterplot of quantity against line revenue should show a fan of points organized into bands, because line revenue is quantity multiplied by a price drawn from a bounded catalog. Predict the structure before drawing it: how many bands, and where should the fan's upper edge sit for a basket of three units?

Code 4.13. Scatterplot of quantity against line revenue

```
lines = transactions_clean.dropna(
    subset=["quantity", "line_revenue"]
)

fig, ax = plt.subplots(figsize=(7, 4))
ax.scatter(lines["quantity"], lines["line_revenue"],
           s=8, alpha=0.3)
ax.set_xlabel("Quantity (units on the line)")
ax.set_ylabel("Line revenue ($)")
ax.set_title("Line revenue against quantity")
plt.show()

# Anything above the catalog ceiling is a specific row that
# owes a specific explanation.
ceiling = lines["quantity"] * 90
strays = lines[lines["line_revenue"] > ceiling]
print("lines above the catalog ceiling:", len(strays))
print(strays[["order_id", "order_line_id", "quantity",
              "unit_price", "line_revenue"]].head(10))
```

Expected output: a fan of points rising with quantity, its upper edge tracking $90 per unit and its lower edge falling below $24 per unit where discounts apply, and — on a correctly cleaned file — zero lines above the catalog ceiling.

Input: the certified line-grain table, restricted to lines with both a quantity and a revenue. Transformation: one point per line, with transparency so that dense bands remain readable. Output: a structure and a test. The structure is the expected one — revenue rising with quantity along bands implied by the $24-to-$90 catalog. The test is the printed stray count, which converts “anything outside the fan is a specific row with a specific explanation owed” from an instruction into a number. A stray above the ceiling at quantity one is an arithmetic defect that survived cleaning, most likely a decimal error of the kind Section 4.5 describes; a cluster of strays sharing a store or a channel is a source-system problem rather than a row problem. Either way the finding goes in the log, and each chart's

question, prediction, and resolution goes in the notebook in words — the written trail is what Chapter 5 will pick up as the difference between an observation and an insight.

4.12 Marketing Interpretation and Managerial Insight

The labs produced a certified table, a completed log, and a first exploration. This section returns to the conference room, because the ten days end in a meeting, and asks what the preparation work means in decision language.

The headline the analytics lead can now deliver is not a number but a warrant for numbers: the table of record reconciles to finance's quarterly total, and every difference between it and the raw extract is a written, defensible decision — this many duplicated lines removed, these erroneous values corrected or excluded under stated rules, these gaps flagged rather than papered over. The reconciliation gap has an explanation instead of a shrug. That warrant changes the texture of the expansion review before a single analysis is presented: the same-age cohort comparison and the store-economics summary will be computed on rows both the CFO and the CRM manager have agreed to trust, which means the meeting can argue about the business instead of about the file — the preparation-stage counterpart of what Chapter 3 said good measurement buys.

Now the wrong managerial reading, because there is a tempting one. A stakeholder scanning the log summary might conclude: “The raw feed overstated revenue; the data team fixed it; so the expansion numbers were too optimistic and the pause should continue.” Every clause outruns the evidence. The duplicates inflated total revenue, but nothing yet says they inflated the suburban stores' revenue disproportionately — duplication that hit all channels evenly changes levels, not comparisons, and it is the comparisons the expansion decision turns on. Nor does a cleaner file settle the direction of any conclusion: the corrected table could make the suburban economics look better (if the removed 999-quantity sentinels had been landing in store rows) or worse. The correction to offer the room is precise: the cleaning changed how much the numbers can be trusted, not yet what the numbers say, and the “what” is next week's work — Chapter 5's — now standing on certified ground. The discipline of separating those two claims is, in miniature, the discipline of the whole course.

Two more managerial translations deserve the room's attention. The first is the judgment entries. The log contains a handful of decisions that were choices, not arithmetic — “web” mapped to Online, missing discounts read as full price with a flag, the impossible-date lines excluded rather than repaired. A manager does not need to review pandas; a manager does need to know that these choices exist, that each has a stated rationale, and that each can be revisited if a stakeholder's knowledge contradicts it. Surfacing them takes three minutes of the meeting and buys something valuable: when a choice is later questioned — and one always is — the answer is a log entry, not a reconstruction from memory.

The second is the language of certification itself. “Certified” is not a synonym for “finished,” and an analyst who uses it loosely will eventually be asked to defend a total that quietly excluded rows nobody mentioned. A table is certified for a stated use, and the statement has a scope: this table supports revenue analysis at order and line grain for this window, with these exclusions, and it does not yet support any analysis that depends on the quantities still unresolved. Part A of the lab makes the point at fourteen rows — eleven rows survive, ten of them carry computable revenue, and the honest label is provisional. Saying so costs a sentence in the meeting. Not saying so costs the analyst's credibility the first time someone recomputes the total and finds a different number.

4.13 Business Analytics in Practice

The chapter so far has practiced preparation on StyleCraft. This section steps out of the fiction and looks at how the same work appears in industry — how much of the job it is, how expensively it fails, and how organizations have tried to industrialize it.

4.13.1 How Much of the Job Preparation Actually Is

The first thing to know is how large this work looms in real analyst time. The folk statistic — that data scientists spend roughly 80 percent of their time finding, cleaning, and organizing data — has circulated for decades, and the precise figure is best treated as a trope rather than a measurement: estimates vary by survey and by decade, and one

widely read practitioner survey put data loading and cleansing at roughly 45 percent of working time rather than 80 (Anaconda, 2020). What the estimates agree on, however, is the ordering: preparation occupies more analyst time than modeling does, a pattern the data-mining literature documented well before modern tooling (Dasu & Johnson, 2003).

What is instructive is how stable the finding has remained across tool generations. Spreadsheets gave way to SQL, SQL to pandas, pandas to AI copilots, and each generation was accompanied by predictions that cleaning drudgery was about to disappear; each generation instead made it faster to produce more data, from more systems, with more inconsistencies. AI assistance is the newest chapter in that story, and the early pattern matches this guide's stance: assistants compress the mechanical hours substantially, while the diagnosis and judgment hours — is this null informative, is this extreme legitimate, which columns define identity — remain stubbornly human, because they were never coding hours in the first place. Analysts who resent preparation as a tax on "real" analytics tend to be surprised by this; analysts who understand preparation as the first analysis, in this chapter's sense, are the ones organizations learn to trust with the file of record.

4.13.2 The Revenue Decline That Was a Tracking Tag

The second vignette is about how silently preparation-level defects can masquerade as business events. A mid-market e-commerce retailer — the composite is generic, but the pattern is one of the most repeated war stories in analytics — opened a quarterly review to a chart showing online revenue in steady decline over six weeks. The room did what rooms do: it generated explanations. A competitor's promotion, a weak product drop, rising ad costs; one executive proposed reallocating budget to paid search that afternoon. The analyst who eventually traced the decline found no customer behavior in it at all: a website tracking-tag update, deployed at exactly the start of the decline, had stopped recording a slice of completed orders, and the "revenue decline" was a measurement outage wearing a trend costume.

The story earns its place in this chapter for two reasons. The defect was invisible to every within-file check — the file was internally consistent, merely incomplete — and would have been caught in a day by the humble discipline of Section 4.3, reconciliation against a known external total (the payment processor's settlement figures never dipped). And the near-miss was a decision, not a dashboard: budget was hours from moving on the strength of a defect. The practice lesson is the chapter's thesis with money attached — an undiagnosed data defect does not merely mislead a chart; it manufactures a business narrative, and rooms act on narratives.

4.13.3 Industrializing the Log

The third vignette concerns industrialization. In organizations that run analytics at scale, the pipeline of Section 4.2 does not live in one analyst's notebook; it lives in engineered data pipelines, run on schedules, with the verification log's logic converted into automated tests. Modern transformation frameworks — dbt is the best-known exemplar of the pattern — let teams attach declarative quality tests to every table in the warehouse: this key is never null, this key is unique, this value stays within an accepted set, this child table's keys all exist in the parent, and whatever additional business rules the team writes as custom tests (dbt Labs, n.d.-a).

Every scheduled run can execute the tests, and organizations may configure a failure to stop a build and alert an owner before a defective table reaches a dashboard. The qualification matters, because the framework supplies the assertions rather than the organizational response: tests carry a severity that determines whether a failure warns or errors, and what happens next — whether a build halts, whether anyone is paged — depends on how the deployment and monitoring around the framework are configured (dbt Labs, n.d.-b). Automation makes the checks cheap and repeatable; someone still has to decide that a failure is worth waking a person for.

Students should recognize the tests immediately — they are Section 4.3's inspection routines, Section 4.5's domain limits, and Section 4.6's cardinality expectations, written once and run forever, and the "known totals" reconciliation of the opening case becomes, in this world, a permanent test rather than a heroic catch. What industrialization does not remove is the analyst's role at the edges of the automation: someone must decide what the tests should assert (the domain limits come from business knowledge, not from the warehouse), someone must adjudicate the failures (is the out-of-band value an error or a policy change?), and someone must own the judgment

entries no test can make. Automation, in other words, industrializes the log's arithmetic and leaves its editorial content exactly where this chapter put it — with the analyst of record.

4.13.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into a single expectation worth internalizing early. You will spend more of your first year preparing data than modeling it; the preparation will periodically be where the real finding lives; and the organization's memory of what was removed, changed, and assumed will reside wherever you write it down — which is to say, if you do not keep the log, it resides nowhere. Cleaning decisions are analytic decisions, and the documentation is not overhead on the work; it is the part of the work that separates the analyst from the liability when a number is challenged six months later. The analysts who advance are rarely the ones who cleaned fastest. They are the ones whose files the organization learned it could certify.

4.14 Ethics, Cleaning Decisions, and the Documented Exclusion

The Business Analytics in Practice section described cleaning as work organizations learn to trust. This section examines the ethical weight of that trust, extending the guide's running discussions (Sections 1.13, 2.12, and 3.13) from data use, problem framing, and measurement design to the cleaning step itself. The chapter's angle is specific: cleaning is silent editorial power, and its ethical discipline is the documented exclusion.

Begin with why the power is editorial. Media scholars have made the point compactly: “raw data” is something of an oxymoron, because data arrives already shaped by collection choices, and every subsequent act of tidying shapes it further (Gitelman, 2013). Digital humanities scholars pressed the argument directly at cleaning, observing that the very word “cleaning” flatters the work — it implies removing grime from a fixed underlying truth, when what actually happens is that someone decides which variations are noise and which are signal, and the decisions are then hidden inside a file that looks pristine (Rawson & Muñoz, 2019). This chapter's vocabulary makes the same point without the theory: the judgment entries in the verification log — what counts as a duplicate, which extremes are errors, what a missing value is assumed to mean, which rows are out of scope — are decisions about whose behavior the certified table represents. They are small, individually defensible, and cumulatively they determine what the expansion review will see.

The ethical risk concentrates where the editing intersects people unevenly. Consider three of the lab's own decisions through that lens. Dropping rows with missing age_band looks hygienic until Section 4.4's mechanism question is asked: if in-store customers skip the optional question more often, the drop disproportionately removes store shoppers — the suburban expansion's own customers — from every age-based analysis, and the review's picture of “who shops the new stores” is quietly reweighted toward people who shop online. Excluding the impossible-date rows is defensible, but the rule must be blind: an exclusion rule applied more diligently to rows that hurt a preferred conclusion than to rows that help it is not cleaning, it is curation. And the treatment of extremes carries a distributional thumb: winsorizing or trimming the right tail of order value specifically shrinks the suburban occasionwear baskets — the strategy's best evidence — which is why Section 4.5 insisted that legitimate extremes are kept and why “remove outliers” is never, by itself, an acceptable log entry. In each case the failure mode is not malice; it is a reasonable-looking default whose burden lands on a group nobody was watching, the same pattern Section 3.6.1 traced for proxy measures (Barocas & Selbst, 2016).

The discipline this guide asks for is therefore procedural, and it has three parts. First, no silent exclusions: every rule that removes or alters rows is written, counted, and carried into the analysis's methods note — “we excluded N order lines under the following three rules” is a sentence every downstream reader is owed. Second, examine what was removed: before certifying, profile the excluded rows as a group — which stores, channels, and customer types they came from — because exclusions that cluster are exclusions with a story, and the story belongs in the log whether or not it is convenient. Third, test the judgment calls that could matter: where a defensible alternative existed (fill missing discounts with 0 versus drop the lines), the flag columns built in Section 4.11 make it cheap to check whether the headline conclusions survive the other choice, and a conclusion that flips on a cleaning judgment is a finding about fragility that decision-makers must hear. None of this slows the work much. All of it converts

editorial power exercised silently into editorial power exercised accountably — which is the only kind an analyst of record should want.

> **CONCEPT**
>
> **The Documented Exclusion**
>
> Every rule that removes or alters rows during preparation must be written as a rule, counted in the verification log, profiled for whom it disproportionately affects, and disclosed wherever the resulting analysis is used. Cleaning exercised without this discipline is silent editorial power over what the organization gets to see; cleaning exercised with it is a defensible analytic decision.
>
> Source: Adapted from Rawson and Muñoz (2019) and Gitelman (2013).

4.15 Chapter Summary

This chapter took StyleCraft from a file that could not be trusted to a table of record, and built the disciplines the trip requires. The main point is that cleaning is the first analysis: every preparation decision — what counts as a duplicate, which extremes are errors, what an absence means, which labels are the same fact, which rows are out of scope — changes the numbers every later chapter will report, and the difference between tidying and professional preparation is that professional preparation makes those decisions explicitly, predicts their effects, and documents them.

The chapter's instrument for that discipline is the verification log, which applies Chapter 1's predict-then-verify to every cleaning step: predicted row counts and totals before, actuals after, every mismatch explained before the pipeline proceeds, and the whole sequence anchored to a known external total. The log ends in a threshold rather than a feeling: a file is certified for a stated use when every prediction has been confirmed or explained and no quantity that use depends on remains unresolved, which is why the lab's miniature closes as provisionally cleaned rather than certified. Around the log, the chapter treated the defect families in turn. Missing values demand a mechanism diagnosis before a remedy, because dropping, imputing, and flagging each carry a nameable bias. Outliers are candidates for investigation rather than removal, detected by IQR fences, z-scores, and domain limits, and classified into legitimate extremes to keep and errors to treat. Duplicates come in exact and near forms, the near form turning on a documented judgment about which columns define an event's identity — and a third form, created by an unvalidated join, that inflates revenue without leaving a trace. Type coercion and categorical standardization make values comparable, with an accounting for everything that fails to convert and for the types the repairs themselves change.

On the constructive side, derived variables and deliberate grain changes through grouped aggregation extend the table's usefulness, verified by the totals an aggregation must preserve and watched for the quieter failure in which an unresolved value is summed into a confident zero. Exploratory data analysis then closes preparation and opens analysis: disciplined questioning of distributions — their skewness and modality — and anomaly-first reading of histograms, box plots, bar charts, and scatterplots. Throughout, AI assistants draft the code while the analyst audits the counts, because the characteristic failures of delegated cleaning — silent row drops, wrong join keys, unstated imputation defaults — are exactly the failures a log catches. The ethics section named the stakes: cleaning is silent editorial power, and the documented exclusion is what makes it accountable.

Looking ahead, StyleCraft now has a certified table and a first look at its shape — including a right-skewed, arguably two-humped order-value distribution that hints at the company's two customer worlds. The next chapter puts the certified data to descriptive work: frequency tables, cross-tabulations, group comparisons, customer profiles, and the applied metric families of revenue, conversion, and retention — including the decomposition that finally explains what the suburban stores' high average order value is actually made of.

4.16 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: diagnose the defect before treating it, predict the effect of every step, keep the log, and read exploratory output for anomalies. They are organized into three groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions on which Assignment #1 draws directly. In-class activities are designed for discussion. Extensions are optional and go beyond the required material. Each exercise also carries its assignment label (required practice, homework submission, in-class discussion, or optional) so that instructors can assign selectively.

4.16.1 Core Chapter Practice

Exercise 4.1 Concept Check (Required Practice)

- State the four stages of the preparation pipeline and the cardinal sin of each.
- What four things does a verification-log entry record before a step is run, and why is a prediction-versus-actual mismatch described in this chapter as “the log working”?
- What does it mean to say a table is certified for a stated use, and what distinguishes a certified table from one that is merely cleaned?
- Explain the difference among MCAR, MAR, and MNAR using age_band in StyleCraft's customers table, and state which remedy-bias pairing each mechanism makes most dangerous.
- Why is “outlier” defined by distance rather than by error, and what evidence distinguishes a legitimate extreme from a genuine error in a transaction file?
- Why does an exact-duplicate check miss a re-exported order, and what judgment must the analyst make to catch it?
- A join to a dimension table can create duplicates that no unmatched-row count will reveal. Explain the mechanism, and name the check that prevents it.
- What does errors=“coerce” do, and what verification must accompany any step that uses it?
- State two totals that must survive a line-to-order grain change exactly, and two reasons a groupby can silently fail to preserve them.
- In one sentence each: what does a histogram detect that describe does not, and what does a box plot show that a pair of group means hides?

Exercise 4.2 Diagnose the Defect (Required Practice)

For each symptom observed during inspection of a transaction extract, name the most likely defect family from Sections 4.4 through 4.7, the routine or check that would confirm the diagnosis, and the first remedy you would consider.

5. describe reports a maximum quantity of 999 while the 99th percentile is 6.
6. The sum of line_revenue is 4 percent above the finance system's figure for the same period.
7. value_counts on a category column returns both “Occasionwear” and “occasion wear.”
8. A revenue column refuses to sum, and dtypes reports it as object.
9. info shows discount_pct with 96 percent non-null values while every other column is complete.
10. A join to the stores table leaves some rows with a missing opening_date.
11. A join to the products table leaves the row count 3 percent higher than it was before the join, and no row is unmatched.
12. A histogram of unit_price shows a lone bar far to the right of the catalog band.
13. Two rows share order_id, customer_id, product_id, quantity, and price but differ in order_line_id and by one day in order_date.

Exercise 4.3 Choose the Missing-Data Remedy (Required Practice)

For each situation, choose drop, impute (state which form), or flag — combinations are allowed — and state in one sentence the bias your choice introduces and why it is acceptable here.

7. 0.5 percent of rows are missing discount_pct; the data dictionary says the modal case is a full-price sale.
8. 12 percent of customers are missing age_band, and the missing share is three times higher among in-store shoppers.
9. campaign_id is null on 40 percent of order lines; the dictionary notes attribution is expected only for campaign-driven orders.
10. A handful of quantity values were set to missing after being classified as sentinel 999s, and the analysis that consumes the table is a revenue total.
11. A per-customer income field, self-reported and optional, is missing for a third of customers, and a colleague suggests filling it with the median.

Exercise 4.4 Predict the Row Count (Required Practice)

A raw extract has 50,000 rows, total revenue of $2,600,000, and inspection has found: 400 exactly duplicated rows; 250 rows forming 125 near-duplicate pairs on the identity columns; 60 rows with order_date before signup_date (exclusion rule agreed); and 900 rows missing discount_pct (to be filled with 0 and flagged). Write the verification log in advance: the predicted row count after each step in the chapter's default order, which steps change total revenue and in which direction, and which steps must leave it untouched. Then state the two questions you would ask before certifying if the actual final row count came back three rows lower than your prediction, and the one additional question you would ask if it came back three rows higher.

Exercise 4.5 Clean the Extract End-to-End (Homework Submission)

Complete Lab 4.1 Part B on the full transactions_raw file: copy the raw frame before the first transformation and keep it to the end, run the pipeline in a documented order, validate the cardinality of all three dimension joins, maintain the verification log throughout (baseline, one entry per step with prediction and actual, judgment entries flagged as such), pass the certification gate on unresolved revenue, reconcile the certified table against transactions_clean and against the known totals, and profile the excluded rows per Section 4.14 (which stores, channels, and customer types they came from). Submit the notebook, the log, and the completed AI-Use Appendix (Appendix D) recording every assistant exchange, including at least one case where the assistant's stated prediction and yours disagreed and how you resolved it.

Exercise 4.6 AI-Assisted Cleaning Audit (Homework Submission)

Give an AI assistant the miniature from Lab 4.1 Part A and this deliberately vague prompt, verbatim: “Clean this dataset and handle the missing values and outliers.” Run its code on a copy. Then audit: reconstruct from the output what the code actually did — rows dropped and under what implicit rule, values imputed and with what default, extremes treated and how, and whether it transformed the raw frame in place — and write a verification-log entry for each step the assistant took, marking every entry that records a decision the assistant made without surfacing it. Conclude with a rewritten prompt that would have forced the choices into the open, and submit the audit with the AI-Use Appendix.

4.16.2 In-Class Activities

Exercise 4.7 Find the Flaw in the Cleaning (In-Class Discussion)

Each scenario below describes a cleaning step that runs without error. Find the flaw, name the section whose discipline it violates, and propose the repair.

9. An analyst removes near-duplicates using the identity columns order_id and product_id only, and the row count drops by more than the near-duplicate inspection predicted.
10. To fix the currency strings, an analyst runs pd.to_numeric with errors=“coerce” and moves on; total revenue falls slightly.
11. An analyst deduplicates first and coerces unit_price afterward, and the O1009-style pair from Lab 4.1 survives into the certified file.

12. An analyst runs every cleaning step directly on the frame loaded from the raw file, then discovers at certification time that the near-duplicate rule was wrong and cannot show what the original labels were.
13. A left join to the products table runs cleanly, every row matches, and total revenue rises by 3 percent.
14. Asked to handle missing ages, an assistant fills them with the overall median; the suburban stores' average customer age falls by four years.
15. An analyst deletes every order above $300 as an outlier because “the catalog tops out at $90.”
16. A groupby to customer grain produces 7,912 rows, but the transactions file contains 8,000 distinct non-null customer_id values and some lines with customer_id missing.
17. An order-grain roll-up reports one order with revenue of exactly $0.00; the line-grain total reconciles perfectly.

Exercise 4.8 Ethics of Exclusion Mini-Cases (In-Class Discussion)

For each mini-case, identify who bears the burden of the cleaning decision, what the documented-exclusion discipline of Section 4.14 requires, and what you would write in the log.

7. Dropping all rows with missing age_band ahead of an age-based targeting analysis, given Exercise 4.3's pattern of in-store missingness.
8. Winsorizing order values at the 95th percentile “to stabilize the averages” in the expansion review.
9. Excluding the newest stores' first eight weeks of transactions as “ramp-up noise” from a store-performance comparison.
10. An exclusion rule for suspect dates that an analyst, short on time, applies only to the store channels because “that's where the problem was.”
11. A colleague proposes deleting the flag columns before circulating the certified file “to keep it tidy.”

4.16.3 Extensions

Exercise 4.9 Reflection Questions (Optional)

Finally, the following questions are intended to be thought-provoking rather than graded.

- This chapter claims that cleaning is the first analysis. What would have to be true for that claim to be wrong?
- Think of a number you have seen reported about an organization you know. What cleaning decisions must have stood behind it, and who would be able to tell you what they were?
- The chapter treats the raw file as evidence. What are the practical limits of that principle when the raw file itself is a vendor's export of something else?
- If an automated test suite can assert everything the verification log asserts, what is left for the log to do — and would you still keep one?
- Where should the line sit between a cleaning decision an analyst may make alone and one that requires a stakeholder's approval? Give an example on each side of your line.

4.17 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Data preparation. The disciplined transformation of raw source data into an analysis-ready table through a repeatable, documented sequence of inspection, cleaning, and verification, such that the result can be reproduced from the untouched raw file (adapted from Dasu & Johnson, 2003; Hellerstein, 2008).

Derived variable. A column computed from existing columns by an explicit, recorded rule at the grain of the table in which it appears — either directly from columns at the current grain or as a field of a coarser table produced by aggregation — documented in the data dictionary and verified against the totals it should preserve (adapted from Kimball & Ross, 2013; Wickham, 2014).

Domain limit. An allowed range or value set declared from business knowledge rather than inferred from the data — the catalog price band, the maximum permitted discount, a plausible basket size — used as an outlier-detection rule and recorded in the data dictionary's allowed-values field (adapted from Hellerstein, 2008).

Duplicate record. A row that records a real-world event already recorded by another row. Exact duplicates are identical in every column; near-duplicates match on the columns that define the event's identity but differ elsewhere, as when an order is re-exported with a new timestamp; and join-created duplicates arise when a merge to a dimension table finds more than one matching row per key.

Exploratory data analysis (EDA). The systematic examination of a dataset's distributions, groups, and relationships through summaries and simple charts, undertaken before formal modeling to discover structure, surface anomalies, and generate questions (adapted from Tukey, 1977; Behrens, 1997).

Flagging. The missing-value remedy that adds an indicator column marking which rows contained an absence, preserving the fact of missingness for later analysis. Flagging records missingness rather than resolving it and is normally combined with dropping or imputation.

Grouped aggregation. The operation, implemented in pandas by groupby, that collects rows sharing a key and summarizes each group, producing a table at a deliberately coarser grain; verified by the totals and counts the grain change must preserve.

Imputation. The replacement of missing values with estimated ones — a central value, the most frequent category, or a model-based prediction — preserving rows at the cost of manufactured data whose bias must be named (adapted from Little & Rubin, 2019; Osborne, 2013).

Missing-data mechanism. The process by which absences arise: missing completely at random (MCAR), missing at random given observed variables (MAR), or missing not at random (MNAR), where missingness depends on the unobserved value itself (Rubin, 1976; Little & Rubin, 2019).

Modality. The number of concentrations, or humps, in a distribution; bimodality often signals that a variable describes a mixture of distinct populations rather than one.

Outlier. An observation unusually far from the bulk of the data by a stated rule (interquartile-range fences, z-score threshold, or domain limits); a candidate for investigation and classification as legitimate extreme or genuine error, not for automatic removal (adapted from Barnett & Lewis, 1994; Tukey, 1977).

Sentinel value. A placeholder a source system records to signal a special condition — such as 999 for an unknown quantity — that masquerades as ordinary data; sentinels are classified as errors and typically converted to missing values with the mechanism documented (adapted from Hellerstein, 2008).

Skewness. The asymmetry of a distribution; right-skewed distributions concentrate values at the low end with a long tail of large values, the chronic shape of marketing quantities such as order value and customer spend.

Type coercion. The conversion of a column from its stored data type to the type its measurement requires, performed with an explicit accounting of every value that fails to convert and of any change the repair makes to what the column claims to measure (adapted from McKinney, 2022; Hellerstein, 2008).

Verification log. The running record, kept alongside cleaning code, in which the analyst predicts each step's effect on checkable quantities (rows, totals, missing counts) before running it, records the actuals after, and explains every mismatch; the chapter's instrument of predict-then-verify, and the basis on which a cleaned file is certified for a stated use once no quantity that use depends on remains unresolved.

Winsorizing. The treatment of extreme values by capping them at a chosen percentile rather than removing them; treated with caution in this guide because it alters real values wholesale and can shrink the decision-relevant tail of a distribution (adapted from Barnett & Lewis, 1994).

4.18 Further Readings

Students who want additional background may begin with the following readings. Cleaning-focused sources are listed first, exploration second.

- Dasu and Johnson (2003) for the classic practitioner treatment of exploratory data mining and cleaning, including the argument that preparation dominates the analytical workload.
- Hellerstein (2008) for a compact, systems-minded survey of quantitative data cleaning — outliers, deduplication, and integrity constraints — that maps closely onto this chapter's defect families.
- Little and Rubin (2019) for the authoritative treatment of missing data; the opening chapters cover the mechanisms at the depth this guide gestures toward.
- Wickham (2014) for the tidy-data principles that explain why some tables are easy to analyze and others fight back.
- Tukey (1977) for the founding text of exploratory data analysis, still bracing; Behrens (1997) for an accessible modern statement of the EDA attitude.
- Rawson and Muñoz (2019) for the argument against treating "cleaning" as neutral — the ethical companion to this chapter's log discipline.

4.19 References

Anaconda. (2020). *2020 state of data science*. https://www.anaconda.com/resources/whitepaper/state-of-data-science-2020

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data* (3rd ed.). Wiley.

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Behrens, J. T. (1997). Principles and procedures of exploratory data analysis. *Psychological Methods, 2*(2), 131–160. https://doi.org/10.1037/1082-989X.2.2.131

Dasu, T., & Johnson, T. (2003). *Exploratory data mining and data cleaning*. Wiley.

dbt Labs. (n.d.-a). *Data tests*. dbt Developer Hub. Retrieved July 12, 2026, from https://docs.getdbt.com/docs/build/data-tests

dbt Labs. (n.d.-b). *Severity*. dbt Developer Hub. Retrieved July 12, 2026, from https://docs.getdbt.com/reference/resource-configs/severity

Gitelman, L. (Ed.). (2013). *"Raw data" is an oxymoron*. MIT Press.

Hellerstein, J. M. (2008). *Quantitative data cleaning for large databases* (Technical report). United Nations Economic Commission for Europe. https://dsf.berkeley.edu/jmh/papers/cleaning-unece.pdf

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

Little, R. J. A., & Rubin, D. B. (2019). *Statistical analysis with missing data* (3rd ed.). Wiley.

Matplotlib Development Team. (2026). *matplotlib.pyplot.boxplot*. Matplotlib documentation. Retrieved July 12, 2026, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html

McKinney, W. (2022). *Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter* (3rd ed.). O'Reilly Media.

Osborne, J. W. (2013). *Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data*. Sage.

pandas development team. (2026a). *Group by: split-apply-combine*. pandas documentation. Retrieved July 12, 2026, from https://pandas.pydata.org/pandas-docs/stable/user_guide/groupby.html

pandas development team. (2026b). *Nullable integer data type*. pandas documentation. Retrieved July 12, 2026, from https://pandas.pydata.org/pandas-docs/stable/user_guide/integer_na.html

pandas development team. (2026c). *pandas.Series.sum*. pandas documentation. Retrieved July 12, 2026, from https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.sum.html

pandas development team. (2026d). *pandas.to_numeric*. pandas documentation. Retrieved July 12, 2026, from https://pandas.pydata.org/docs/reference/api/pandas.to_numeric.html

Rawson, K., & Muñoz, T. (2019). Against cleaning. In M. K. Gold & L. F. Klein (Eds.), *Debates in the digital humanities 2019* (pp. 279–292). University of Minnesota Press.

Rubin, D. B. (1976). Inference and missing data. *Biometrika, 63*(3), 581–592. https://doi.org/10.1093/biomet/63.3.581

Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley.

Wickham, H. (2014). Tidy data. *Journal of Statistical Software, 59*(10), 1–23. https://doi.org/10.18637/jss.v059.i10

Chapter 05

Descriptive Analytics and Customer Insights

What the Average Is Made Of: Counting, Comparing, and Decomposing on Certified Data

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops descriptive analytics as the discipline of summarizing certified data without distorting it. It treats counting and the denominator problem, the choice of a measure of center against distribution shape, and spread as a finding in its own right. Its structural core is the cross-tabulation: three percentage bases, the direction rule, and Simpson's paradox computed rather than merely cited. The chapter then assembles the applied marketing metric families and the decomposition identity, which factors a composite metric such as average order value exactly into basket size and revenue per unit. Cohort retention tables and campaign funnels extend description along time and along the customer journey; concentration analysis, benchmarking, and the insight statement close the sequence. The labs decompose StyleCraft's suburban-versus-urban average-order-value gap and build profiles, cohorts, concentration, and campaign metrics on the certified file.

KEYWORDS

descriptive analytics; frequency tables; cross-tabulation; simpson's paradox; average order value; cohort analysis; conversion funnel; concentration analysis; customer insight; marketing analytics

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Descriptive analytics and customer insight. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 5, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors,

researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

16. Produce and interpret frequency tables, proportions, summary statistics, and cross-tabulations from transaction and customer data.
17. State the denominator problem and, for any reported proportion or rate, identify the population it counts over and the populations it silently excludes.
18. Choose the measure of center that fits a distribution's shape, and defend the choice of mean, median, or mode for a skewed marketing quantity.
19. Report spread alongside center — range, interquartile range, and standard deviation — and recognize heterogeneity as a finding rather than noise.
20. Read a cross-tabulation three ways, compute row and column percentages correctly, and detect Simpson's paradox by checking group composition before trusting a comparison of rates.
21. Build customer profiles and group comparisons on distributions, not just on means, from marketing data.
22. Summarize revenue, acquisition, conversion, retention, and engagement metrics at stated grains, and verify every summary against known certified totals.
23. Decompose a composite metric exactly into its factors — including average order value into basket size and revenue per unit — and verify that the factors multiply back.
24. Construct and read a period-active cohort retention table and a descriptive campaign funnel, and explain what cohort alignment does and does not remove from a comparison.
25. Quantify concentration with deciles and Pareto shares, place numbers in context with benchmarks, and turn descriptive findings into insight statements written in managerial language — while keeping descriptive language descriptive.

Chapter 4 ended with two things in hand: a certified table of record, reconciled to finance's known total with every departure from the raw file logged, and a first exploratory look that turned up a right-skewed, arguably two-humped distribution of order values — a shape that hinted at two customer worlds living inside one company. Chapter 4 also made a promise: that the next chapter would finally explain what the suburban stores' celebrated average order value is actually made of. This chapter keeps that promise. It develops descriptive analytics as a precision discipline — counting, summarizing, cross-tabulating, profiling, and decomposing — and shows that the most consequential analytical work at most companies is exactly this work, done carefully and narrated honestly. The destination is customer insight: descriptive findings translated into statements a decision-maker can act on without being misled by them.

CONCEPT

What This Chapter Is Really About

Chapter 4 argued that cleaning is the first analysis. This chapter makes the companion argument for the step that follows: summarizing is the second analysis, and it is nowhere near as innocent as it looks. Every descriptive choice — which denominator, which measure of center, which grouping, which percentage direction, which comparison — changes the story a certified table tells, without changing a single number in the table. The suburban stores' high average order value is a true number; whether it means the expansion is working depends entirely on what that average is made of and who is underneath it. An undisciplined analyst reports the number; a disciplined analyst decomposes it, reconciles every summary to the certified totals, reports the spread with the center, and writes insight statements that say exactly as much as the description supports and no more. In other words, description is not the warm-up before the real analytics. Done precisely and narrated well, description is most of the analytics value a business ever receives.

5.1 Marketing Decision Context: The Meeting About One Number

Chapter 4 closed with a quiet institutional achievement: ten days of disciplined preparation produced a certified table of record, reconciled to finance's quarterly total, with a verification log standing behind every row that was removed or changed. The quarterly expansion review can now argue about the business instead of arguing about the file. This chapter is that argument.

The review convenes in seven days, and the Chapter 2 memo committed the analytics team to two deliverables for it: a same-age cohort comparison of the new suburban stores against the urban stores at the same age, and a store-economics summary. Both are, in the vocabulary of Section 1.4, descriptive work — no model, no forecast, no experiment. Just counts, averages, rates, and comparisons, computed on the certified table. On paper, the easy week.

Then the pre-read goes out, and one number swallows the meeting before it starts. The finance team's summary shows that the average order value at the suburban and resort stores runs far above the average order value online and at the urban stores — not by a rounding margin, but by a multiple. Within a day, the number has acquired a life of its own. The head of retail circulates it with a one-line message: proof the affluent-suburban strategy is working, and grounds to unpause the next wave of store openings. The CFO reads the same number more cautiously and asks the question that gives this chapter its work: catalog prices top out near ninety dollars, the same weekly drops ship to every store, and nothing at Greenwich costs more than it costs in SoHo — so what, exactly, is that average made of? And the CRM manager, whose worry about declining repeat purchase opened this guide back in Chapter 1, adds the uncomfortable companion question: an average is computed over orders — what does it say, if anything, about the customers placing them?

The decision at stake is capital allocation: whether the review recommends resuming the suburban store pipeline, and the pre-read number is on track to carry that decision by itself. The analytics lead's task is therefore not to produce more numbers but to make the numbers already on the table mean something defensible. That requires answering three questions in descriptive language. First, the composition question: the average-order-value gap must be decomposed — since unit prices cannot explain it, is it larger baskets, a different product mix, or both, and by what measurable factors? Second, the people question: averages summarize orders and customers into single figures — who is underneath the suburban average, how much do they vary, and is the healthy-looking mean hiding an unhealthy distribution? Third, the comparison question: every comparison in the pre-read pools stores, channels, and customer vintages of very different compositions — do the headline comparisons survive when the groups are made comparable, or is the review about to be steered by a mix effect wearing a trend costume?

The deadline is fixed and the audience is not technical. Whatever the analytics lead finds must arrive as clear tables and short sentences: this is what the number is made of, this is who is underneath it, this is what it does and does not support. The chapter builds that toolkit in order. Sections 5.2 through 5.4 establish precise counting and

honest summarizing — frequency tables and the denominator problem, the choice of center, and spread as a finding. Sections 5.5 and 5.6 build the comparison machinery: cross-tabulations, the three percentages, Simpson's paradox, and group profiles. Sections 5.7 through 5.9 assemble the applied metric families, the decomposition identity that answers the CFO's question, cohorts and funnels, and the concentration and benchmarking context that turns numbers into claims. Section 5.10 examines how AI assistants help and fail at exactly this work, and the labs in Section 5.11 perform the decomposition on StyleCraft's certified data — where the answer, by design, is checkable.

5.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

14. The suburban average order value is a true number computed from certified data. List three different decisions a stakeholder might take from it, and what additional descriptive fact each decision silently assumes.
15. Catalog prices top out near $90 in every store. If unit prices cannot explain the average-order-value gap, what two quantities could? Write the arithmetic identity that connects them to average order value before reading Section 5.7.
16. The CRM manager objects that an average over orders says nothing about customers. What single change of grain (Section 3.3) turns the order-level average into a customer-level claim, and what new denominator does it require?
17. The pre-read compares all suburban stores against all urban stores. The stores opened in different waves and their customers differ in tenure. Which of this chapter's tools protects that comparison, and from what?

5.2 Counting Done Right: Frequency Tables, Proportions, and the Denominator Problem

The opening case turns on the meaning of one average, but averages are built from counts, so the chapter begins where description begins: counting things and stating shares. This looks beneath a graduate course's dignity until the first dispute, which in practice arrives within minutes of the first table.

A frequency table is the simplest descriptive object there is: the distinct values of one variable, and how many observations take each value. Run against the certified transactions table, a frequency table of channel answers “how many order lines came through Online, App, and Store”; run against customers, a frequency table of loyalty_tier answers “how many customers sit in each tier.” The count column is rarely the useful one, however. Decision language runs on shares, so the working form of the frequency table adds a proportion column — each count divided by the total — and the moment a proportion appears, so does the chapter's first discipline: the total it is divided by must be named.

DEFINITION

Frequency Table

A frequency table is a summary that lists each distinct value (or binned range) of a variable together with the count of observations taking that value and, in its working form, each count's proportion of a stated total. A frequency table is complete only when its unit of analysis and its denominator — what one row of the underlying data is, and what the proportions are shares of — are stated alongside it.

Source: Adapted from Freedman et al. (2007) and Agresti (2019).

Table 5.1 shows the working form on an illustrative extract of one thousand order lines, and it already contains a small trap worth pausing on. The proportions in the table are shares of order lines, because the underlying data sits at the order-line grain established in Section 3.3. They are not shares of orders (an order can span several lines), not shares of revenue (lines carry different values), and not shares of customers (a frequent customer contributes many lines). All four statements — share of lines, of orders, of revenue, of customers — are legitimate descriptive claims,

they will generally all differ, and each answers a different business question. “Store accounts for 44 percent of order lines” and “Store accounts for 31 percent of customers” can both be true of the same file at the same time.

Table 5.1

A frequency table in working form (illustrative extract, 1,000 order lines)

Channel	Order lines	Share of lines
Online	460	46.0%
Store	440	44.0%
App	100	10.0%
Total	1,000	100.0%

This is the denominator problem, and it deserves its name because it is the single most common way a correct count becomes a wrong claim. Chapter 3 established that a complete ratio metric names its numerator, denominator, window, and filters (Section 3.5.1); the denominator problem is what happens in descriptive work when that discipline lapses — a share is reported, the reader supplies a denominator from imagination, and the imagined denominator is not the one used. An email “open rate of 38 percent” means one thing per email delivered and another per email sent; a “repeat purchase rate” moves by several points depending on which customers the denominator admits, which is exactly the dispute Chapter 3's lab staged (Section 3.10.3); and the pre-read's average order value is revenue per order — a denominator choice with everything riding on it, as Section 5.7 will show. The remedy is not sophistication but habit: every proportion, rate, and average travels with its denominator, in the table note or in the sentence, every time.

CONCEPT

The Denominator Problem

Any count can be turned into several different proportions, each correct over its own denominator and each supporting a different claim. A share is not a fact about the numerator; it is a fact about a numerator-denominator pair. Never report or accept a proportion, rate, or average without naming what it is divided by — and when two stakeholders quote different values for the “same” rate, check the denominators before checking the arithmetic (see Section 3.5.1 for the full metric-definition discipline).

Moreover, counting done right respects one more habit inherited from Chapter 4: the frequency table's total row is a verification instrument, not decoration. The counts in Table 5.1 must sum to the certified extract's row count, and the proportions to 100 percent; a frequency table of the full transactions file whose total disagrees with the verification log's certified row count is announcing that a filter ran silently somewhere. This chapter will make that reconciliation habit systematic in Section 5.7, but it starts here: the first thing to check about any summary is that it still adds up to the file it summarizes. With counting and shares established, the next section turns to the summary that dominates every marketing conversation — the average — and to the choice hiding inside it.

5.3 The Center That Fits the Shape: Mean, Median, and Mode

Section 5.2 counted values one variable at a time. The natural next step is to collapse a numeric variable into a single representative figure — a measure of central tendency — and here description makes its first genuinely consequential choice, because there are three standard centers and skewed data makes them disagree.

The three candidates are familiar. The mean is the arithmetic average: the sum of the values divided by their count. The median is the middle value: the point with half the observations below and half above. The mode is the most frequent value, or for binned data the tallest bar — the peak of the distribution. On many symmetric, unimodal distributions the three sit close together or coincide, and the choice among them is nearly cosmetic. Marketing data

is rarely symmetric. As Section 4.9.1 established, quantities such as order value, customer spend, and engagement counts are chronically right-skewed: most values are modest, and a long tail of large values stretches to the right. On such a shape the three centers pull apart, and on the right-skewed distributions marketing analysts meet most often they fall in a familiar order — the mode lowest at the peak, the median above it, and the mean highest, dragged toward the tail. That ordering is a dependable working heuristic rather than a theorem; the exact arrangement is not guaranteed for every distribution, and what does hold wherever the mean exceeds the median is the consequence that matters: each center is now answering a different question.

DEFINITION

Central Tendency

Measures of central tendency summarize a numeric variable with a single representative value: the mean (the arithmetic average, sensitive to every value and therefore to extreme ones), the median (the middle observation, resistant to extreme values), and the mode (the most frequent value or tallest bin, the only center defined for categorical data). On skewed distributions the three diverge, and the choice among them is an analytic decision, not a formatting preference.

Source: Adapted from Freedman et al. (2007).

The choice discipline is best stated as questions. If the business question concerns totals — revenue, budget, capacity — the mean is the right center, because the mean is the total in disguise: mean order value times order count is total revenue, exactly, which is why finance thinks in means. If the question concerns the typical case — what does a normal order look like, what will the typical customer spend — the median serves better on skewed data, because it refuses to let a handful of large baskets speak for the many small ones. And if the question concerns the most common case — the price point most units actually transact at, the most frequent basket size — the mode answers it, and is additionally the only center available for categorical variables like channel or loyalty tier. Table 5.2 summarizes the mapping. The professional habit this guide asks for is stricter than choosing well: on skewed data, report mean and median together, because the gap between them is itself descriptive information — it tells the reader, in one comparison, that a tail exists and how hard it pulls.

Table 5.2

Choosing the measure of center

The question being asked	Right center	Why
Totals: revenue, budget, capacity	Mean	The mean times the count reproduces the total exactly
The typical case on skewed data	Median	Resistant to the long tail; speaks for the middle observation
The most common case; categorical data	Mode	The peak of the distribution; the only center for categories
Skewed data, any audience	Mean and median together	Their gap discloses the tail's pull

One computational trap deserves a boxed warning of its own, because it recurs from Chapter 1's exercises to the executive pre-read: the averaged average. The mean of group means is not, in general, the overall mean; the overall mean is the weighted mean, each group's mean weighted by its size. Averaging StyleCraft's per-store average order values gives every store an equal vote regardless of how many orders it processed, and the resulting figure reconciles to no total at all. The pooled figure — total revenue divided by total orders — is the one that adds up. Exercise 1.10, item 2, “The Averaged Average,” previewed this trap on conversion rates (the simple average of four campaign rates versus the pooled rate); it returns here as a standing rule, and it returns once more in this chapter's labs as a verification check, because the equality “weighted recombination of the parts equals the certified whole” is exactly the kind of known-totals check this chapter's verification theme is built on.

In other words, the average is not a neutral summary. It is a choice among centers, made against a shape the analyst is obliged to have looked at (Section 4.9.1), reported with its companion center when the shape is skewed, and recombined by weights, never by simple averaging, when groups are involved. What the average deliberately discards — how much the values vary around it — is the subject of the next section, and discarding it is not always acceptable.

5.4 Spread Is a Finding: Range, IQR, Standard Deviation, and Heterogeneity

Section 5.3 compressed a variable to one number. This section argues that the compression loses something a decision-maker often needs, and introduces the measures that recover it: measures of spread, or dispersion. The section's larger claim is cultural as much as technical — in marketing data, heterogeneity is not noise around the real answer; heterogeneity frequently is the answer.

Three measures cover the practical ground. The range — maximum minus minimum — is the bluntest: easy to state, ruined by a single extreme value, and mostly useful in inspection (a range check is how Chapter 4 caught the out-of-band prices). The interquartile range (IQR) — the distance between the 25th and 75th percentiles — describes the width of the middle half of the data and inherits the median's resistance to extremes; readers of Chapter 4 have already used it structurally, since the IQR is the box in a box plot and the basis of the outlier fences of Section 4.5. The standard deviation (SD) summarizes the typical distance of observations from the mean; it uses every value, which makes it informative on well-behaved data and tail-sensitive on skewed data, where it should be read alongside the IQR rather than instead of it.

DEFINITION

Dispersion

Measures of dispersion summarize how much a numeric variable varies: the range (maximum minus minimum, maximally sensitive to extremes), the interquartile range (the width of the middle 50 percent of observations, resistant to extremes), and the standard deviation (the typical distance from the mean, computed from every observation). A center reported without a companion measure of spread is an incomplete description.

Source: Adapted from Freedman et al. (2007) and Tukey (1977).

Why insist on spread in a managerial document? Because two groups with identical means can demand opposite marketing treatments. Consider two stores, each with a mean order value of sixty dollars. In the first, orders cluster tightly between fifty and seventy: one customer story, one merchandising story, one number that genuinely represents its orders. In the second, the same mean arises from many thirty-dollar baskets and a steady stream of ninety-dollar occasionwear baskets: two customer stories sharing a checkout line, and a mean that represents neither. Every summary statistic in a pre-read looks like the first store; only the spread — and, for the sharpest cases, the distribution's shape — reveals the second. This is precisely the situation Chapter 4's exploration hinted at for StyleCraft as a whole: a two-humped order-value distribution (Section 4.9.1) is bimodality, and bimodality is the distribution announcing that it describes a mixture of populations. A mean computed across a mixture is an average of nobody.

CONCEPT

Heterogeneity Is a Finding

When a group's members vary widely — in spend, frequency, basket composition, or engagement — that variation is itself a descriptive result, often more decision-relevant than the group's average. Report spread with every center; inspect shape (Section 4.9.1) before trusting any center; and treat wide or two-humped dispersion as a prompt to split the group and describe the parts, not as noise to be averaged away. The question “how much do they vary?” is the question “is this really one group?” wearing statistical clothes.

The practical reporting standard, then, is a small package rather than a single figure: a center chosen for the shape, a second center when skew makes it informative, a spread measure suited to the audience (the IQR travels well because “the middle half of orders fell between X and Y” is plain English), and the group's size — because a mean over forty orders and a mean over forty thousand deserve very different confidence, a point Section 5.9 returns to under benchmarking. This package is what Section 5.6 will assemble into profiles.

Finally, the forward significance of this section should be named, because it shapes the whole book. Heterogeneity as a finding is the descriptive doorway to Part II: if StyleCraft's customers are not one population, the natural next question is which populations they are, and answering it formally is segmentation — Chapter 6's subject. This chapter stays descriptive: it will split groups by known dimensions (metro, channel, tier) and describe the parts. Discovering unknown groups is next month's work; noticing that the average hides them is this month's.

5.5 Cross-Tabulations: Two Variables, Three Percentages, and Simpson's Paradox

The tools so far describe one variable at a time. Marketing questions are almost never about one variable: they are about relationships between categories — channel by metro, tier by age band, repeat purchase by store type. The descriptive instrument for two categorical variables is the cross-tabulation, and it is this chapter's structural core, because both the power and the danger of the expansion pre-read live inside it.

A cross-tabulation (or contingency table) counts observations for every combination of two categorical variables: rows for the values of one, columns for the values of the other, a count in each cell, and totals on the margins. Table 5.3 shows an illustrative extract crossing customer home metro against primary channel. As with the frequency table, the counts are only the beginning; the analytical content arrives when counts become percentages — and a cross-tabulation supports three different percentage bases, which support three different claims. Take the single cell holding the 540 suburban store-primary customers. Cell percentages divide by the grand total: 540 ÷ 2,000, so 27.0 percent of all customers are suburban store-primary. Row percentages divide by the row total: 540 ÷ 800, so 67.5 percent of suburban customers are store-primary. Column percentages divide by the column total: 540 ÷ 840, so 64.3 percent of store-primary customers are suburban. One cell, three correct percentages, three different sentences — and reading one as another is among the most common table-reading errors in professional life.

Table 5.3

A cross-tabulation read three ways (illustrative extract, 2,000 customers)

Home metro	Online/App primary	Store primary	Row total
NYC Urban	900	300	1,200
NYC Suburban	260	540	800
Column total	1,160	840	2,000

Which percentage should be reported? The classic guidance is directional: percentage in the direction of the explanatory variable — compute percentages within the groups you are comparing, so that the comparison reads

across groups on a common base (Zeisel, 1985). To ask “do suburban customers shop differently from urban customers?” is to compare metros, so the percentages run within each metro row: 75.0 percent of urban customers are online/app-primary against 32.5 percent of suburban customers, and the behavioral contrast is stated cleanly. The column percentages answer a different, equally legitimate question — “who is the store channel serving?” — and belong to conversations about channel investment, not customer behavior. The discipline is to decide which question is being asked, percentage in that direction, and label the table so a reader cannot mistake the base: a cross-tabulation without labeled percentage bases is the denominator problem of Section 5.2 in two dimensions.

Now the danger. Because a cross-tabulation compares rates across groups, it inherits the deepest trap in descriptive comparison: the groups being compared may differ in composition, and composition can reverse a comparison outright. Section 3.3.1 established the general principle — aggregation changes the claim, and a relationship at one level of analysis need not hold at another. Simpson's paradox is that principle's table-level case, and this chapter's obligation, having cited the principle, is to compute it (Simpson, 1951; Blyth, 1972).

DEFINITION

Simpson's Paradox

Simpson's paradox is the reversal or disappearance of an association between two variables when the data are disaggregated by a third: a group can show the higher rate in every stratum yet the lower rate in the aggregate, because the groups are unevenly distributed across strata with very different base rates. The paradox is arithmetic, not error — the aggregate and the strata are all computed correctly — which is why it is detected by examining composition, not by rechecking sums.

Source: Adapted from Simpson (1951) and Blyth (1972).

Work the arithmetic on numbers built for clarity. Suppose an analyst compares repeat-purchase rates by primary channel and finds, in aggregate, that store-primary customers repeat at 42.5 percent (357 of 840) while online-primary customers repeat at 52.2 percent (605 of 1,160) — apparently a 9.7-point indictment of the store experience. Table 5.4 disaggregates the same two thousand customers by home metro, and the indictment reverses: within the urban metro, store-primary customers repeat at 65 percent versus 60 percent online; within the suburban metro, 30 percent versus 25 percent. Store-primary customers repeat at a higher rate in every metro and a lower rate overall. The resolution is composition: store-primary customers are concentrated in the suburban metro, where everyone — online or store — repeats far less, so the store column inherits the suburban base rate. The aggregate comparison is not wrong arithmetic; it is a channel comparison contaminated by a metro comparison, and the contaminant is exactly the designed StyleCraft tension this book keeps meeting: the suburban expansion attracts customers who buy richly and return rarely.

Table 5.4

Simpson's paradox in repeat-purchase rates (illustrative extract, 2,000 customers)

Group	Online/App primary	Store primary
NYC Urban: repeaters / customers	540 / 900 (60.0%)	195 / 300 (65.0%)
NYC Suburban: repeaters / customers	65 / 260 (25.0%)	162 / 540 (30.0%)
Aggregate: repeaters / customers	605 / 1,160 (52.2%)	357 / 840 (42.5%)

Note what the aggregate row does and does not say. The 9.7-point aggregate gap favors the online channel; every stratum favors the store channel by five points. Nothing was computed wrongly in either direction, and an analyst who reports only one of the two rows has chosen a conclusion rather than described a table.

Three working rules follow. First, before trusting any comparison of rates across groups, check the composition of the groups on the variables that plausibly drive the rate — a one-line cross-tab of group by stratum is usually enough to see the imbalance. Second, when composition differs, report the stratified comparison and say why: “within each metro, store-primary customers repeat slightly more; the aggregate reverses because store customers

skew suburban" is one sentence and it is the honest one. Third, resist the promotion of either table to a causal story: neither the aggregate nor the strata say that the store channel causes anything — confounding and its formal treatment are Chapter 7's subject, and the descriptive analyst's job ends at an accurate, composition-aware account of what the rates are. Where does that leave the pre-read? Every headline comparison in it — suburban versus urban, new stores versus old — pools groups of different composition, and Section 5.8's cohort tables are the standing defense. First, though, the comparison toolkit needs its general form: profiling groups on many descriptives at once.

5.6 Group Comparisons and Customer Profiling

The cross-tabulation compares groups on one categorical outcome. Real stakeholder questions ask for more: not just "do suburban customers repeat less" but "who are these customers?" The descriptive answer is the profile — and this section's discipline is that groups are compared on distributions, not on lone averages.

A group comparison on a numeric variable should, by Sections 5.3 and 5.4, travel as a package: each group's size, a shape-appropriate center (mean and median together when skewed), and a spread measure. Side-by-side five-number summaries — minimum, first quartile, median, third quartile, maximum — do this compactly, and their visual form is the grouped box plot whose reading Chapter 4 established (Section 4.9.2); this chapter adds only the comparative habit of putting the groups on one axis and reading widths and overlaps, not just medians. Grouped box plots disclose differences in medians, in spread, in overlap, and in the tails, and those differences are how an analyst identifies the comparisons worth investigating. What overlap alone does not establish is whether a difference is statistically reliable or operationally important. Two groups whose boxes barely overlap are a strong lead, not a proven distinction; two groups whose means differ while their boxes coincide almost entirely may matter to finance and not at all to a campaign planner deciding whom to treat differently. The picture nominates comparisons; the decision context, and eventually the evidence standards of Chapters 7 and 11, adjudicate them.

A customer profile assembles such comparisons into one table: rows for the groups being compared, columns for a chosen set of descriptives, every cell computed at a stated grain. Table 5.5 sketches the form for the expansion review's central comparison. Note the grain discipline the table forces: order-level facts (average order value, units per order) and customer-level facts (orders per customer, repeat rate, revenue per customer) coexist in one profile, so each column must declare its unit of analysis, and building the customer-level columns requires exactly the deliberate grain change — transactions grouped to customers — that Section 3.3 defined and Chapter 4's lab practiced (Section 4.8).

Table 5.5

A customer profile at two grains (structure; values computed in Lab 5.2)

Descriptive (grain)	NYC Urban	NYC Suburban
Customers (customer)	—	—
Orders per customer, mean / median (customer)	—	—
Repeat-purchase rate, headline definition per Section 3.10.3 (customer)	—	—
Average order value (order)	—	—
Units per order, mean (order)	—	—
Occasionwear share of units (order line)	—	—
Revenue per customer, mean / median (customer)	—	—

CONCEPT

A Profile Is an Argument

A profile is not a data dump; it is a curated set of comparisons chosen to answer the analytic specification (Section 2.5). Every row earns its place by bearing on the decision; every cell declares its grain and denominator; skewed columns carry median beside mean; and group sizes appear first, because every other number in the table is only as steady as the count behind it. Two analysts profiling the same groups for different decisions should produce different tables — and both should reconcile to the same certified totals.

The profile's power is that it converts an adjective into arithmetic. “The suburban shopper is older, more affluent, less frequent, and occasion-driven” is the strategy deck's sentence; the profile is where each clause acquires a number, a denominator, and a spread — or fails to. Its risk is the same as every comparison in this chapter: a profile of pooled groups inherits every composition problem of Section 5.5, so profiles of store groups with different opening dates or customer vintages must either stratify (profile within cohorts, per Section 5.8) or disclose the imbalance in a note. The labs enforce both habits. With the comparison machinery built, the chapter now turns to the specific quantities marketing profiles are made of — the applied metric families — and to the identity that finally answers the CFO's question.

5.7 The Applied Metric Families

The descriptive machinery of Sections 5.2 through 5.6 is general. Marketing runs it over a recurring repertoire of quantities, and fluency in that repertoire — what the standard metrics are, what denominators define them, and how they compose — is part of the job description (Farris et al., 2010). This section organizes the repertoire into five families and introduces the decomposition identity, the chapter's central analytic move. One boundary first: this section describes the families and their watchpoints; the complete formula catalog lives in Appendix B and only there (Section 3.8.1's rule that metrics belong in the dictionary applies to the guide itself), and every rate below is a ratio metric owing the full four-element definition of Section 3.5.1.

The revenue family summarizes what customers spend. Its workhorses are average order value (AOV) — total revenue divided by order count, an order-grain mean — and revenue per customer, a customer-grain mean over a stated window. The two are routinely conflated and must not be: AOV knows nothing about frequency, so a store can lead on AOV and trail badly on revenue per customer, which is precisely the CRM manager's objection in the opening case. The acquisition family prices the top of the relationship: customer acquisition cost (CAC) — acquisition spend divided by customers acquired, with the window and the definition of “acquired” doing heavy work — and cost per click (CPC), spend divided by clicks. The conversion family describes movement through stages: click-through rate (whose construction Chapter 1 previewed in Section 1.10), and the stage-to-stage rates Section 5.8 will structure. The retention family describes continuation: the repeat-purchase rate — whose three-definitions dispute Chapter 3 staged and settled with a headline definition (Section 3.10.3) — and its complement in spirit, the descriptive churn rate, the share of a stated customer population inactive by a stated rule over a stated window. Churn here is purely descriptive bookkeeping: predicting which individual customers will churn is Chapter 9's subject, and nothing in this chapter models anything. Finally, the engagement family covers the permission channels: open rate and click rate for email and SMS, where the denominator problem is at its most treacherous (sent, delivered, and unique-opened denominators all circulate) and where a boxed reminder from Section 5.2 applies with full force.

Table 5.6

The applied metric families (formulas cataloged in Appendix B)

Family	Representative metrics	Grain of the claim	Denominator watchpoint
Revenue	AOV; revenue per customer	Order; customer	Per order and per customer differ by frequency — never interchange them
Acquisition	CAC; CPC	Cohort of acquired customers; click	What counts as "acquired," and over what window
Conversion	CTR (Section 1.10); stage rates	Impression; stage population	Rate of the step versus rate from the top, and whether the stages share a unit (Section 5.8)
Retention	Repeat-purchase rate (Section 3.10.3); descriptive churn rate	Customer	Which customers the denominator admits, and the window
Engagement	Open rate; click rate	Message or recipient	Sent versus delivered versus unique — name the base

Now the identity. Many headline metrics are composites — products of simpler factors — and a composite can be decomposed exactly: split into factors that multiply back to the original, so that a gap in the headline becomes the product of gaps in its factors. The decomposition is multiplicative, and saying so plainly prevents the most common misuse of it: the identity yields *factors*, not additive percentage shares. A sentence of the form "basket size accounts for sixty percent of the gap" claims an allocation the identity does not supply, and supplying one requires an additional convention (a logarithmic contribution decomposition, for instance) that this chapter does not need. For the review's number, the identity is short. Revenue over orders equals (units over orders) times (revenue over units): AOV = units per order × revenue per unit.

Average order value is basket size times average realized price per unit — an identity, true by construction, at every grain and in every group. And the second factor decomposes once more, because average price per unit in a catalog with stable list prices is driven largely by mix: revenue per unit equals the sum, across product categories, of each category's share of units times its average realized unit price. Together the two steps answer the CFO's question in the only way arithmetic can: if suburban AOV exceeds urban AOV while list prices are consistent across stores, the difference must arise from basket size, realized revenue per unit, or both — and the decomposition states the factor each contributes, exactly. The realized-price factor may in turn reflect product and category mix (occasionwear at the top of the $24-to-$90 band), discount depth, promotional participation, or, depending on how revenue is defined, returns and adjustments; which of those is doing the work is a second question, and Section 5.11 shows how to ask it. The extension to the CRM manager's question is one more factor: revenue per customer equals AOV times orders per customer, which is where a weak repeat rate quietly taxes a strong basket.

CONCEPT

The Decomposition Identity and the Known-Totals Check

A composite metric is decomposed by writing it as a product of factors that cancel exactly (revenue/orders = units/orders × revenue/units), computing each factor per group, and verifying two equalities before any interpretation: the factors must multiply back to the composite in every group, and the group summaries must recombine — weighted, never simply averaged (Section 5.3) — to the certified totals of Chapter 4's verification log. A decomposition that does not multiply back has a computation error; a summary that does not recombine to the certified total has a silent filter.

Report factors, not shares. The identity produces multiplicative factors whose product is the composite; it does not apportion the gap into percentages, and a percentage claim requires a stated allocation convention the identity alone does not provide. This is the chapter's expression of predict-then-verify (Section 1.7): the answers are checkable, so check them.

The identity does one more quiet service: it disciplines interpretation. “Suburban AOV is higher because suburban customers are wealthier” is a story; “suburban AOV is 2.5 times urban AOV, factoring into a 1.67× basket-size factor and a 1.50× realized-price-per-unit factor, with the price factor tracking category mix rather than list prices” is a description — auditable, decomposable, and silent about causes it cannot see. The labs compute exactly this on StyleCraft's certified data, where the decomposition is a designed finding the dataset ships with, so students can grade themselves. Before the labs, two more descriptive structures complete the toolkit: description along time, and description along the journey.

5.8 Cohorts and Funnels: Description Along Time and Along the Journey

Sections 5.5 and 5.6 warned repeatedly that pooled comparisons inherit composition problems. This section builds the two descriptive structures that manage composition by construction: the cohort, which holds customer vintage fixed, and the funnel, which holds the journey's stages in order.

Begin with the cohort. Comparisons of retention across stores or channels are chronically contaminated by tenure: a store opened eight months ago cannot have two-year-loyal customers, so pooling its customers with a flagship's and comparing repeat rates compares store quality and customer age at once — the Simpson's structure of Section 5.5 with time as the lurking stratum. The fix is to compare like vintages with like.

DEFINITION

Cohort

A cohort is a group of units defined by a shared starting event in a shared time window — for StyleCraft, typically all customers whose first purchase (or signup) fell in a given month or quarter. Cohort analysis follows each cohort forward from its own starting point, so that groups are compared at the same age rather than at the same calendar date.

Aligning on age removes tenure from the comparison and makes vintage differences visible. It does not make the comparison clean. Two cohorts reach a given age in different calendar months, so seasonality, campaigns, pricing changes, and economic conditions can still differ between them, and cohorts acquired through different channels differ in composition from the start. Cohort alignment reduces tenure-driven mix bias; it does not eliminate period or compositional effects.

Source: Adapted from Glenn (2005).

The cohort's standard descriptive artifact is the cohort retention table: one row per cohort, one column per period since the starting event, and in each cell a measure of how much of the cohort is still there. Two structures answer to that description, and a table that does not say which it is cannot be read. In a *period-active* table, a cell holds the share of the cohort that was active — by a stated activity rule — during that individual period; because a customer can be inactive in Month 2 and return in Month 3, a period-active rate can rise as well as fall. In a *survival retention* table, a cell holds the share retained continuously through that period under a stated rule, and survival rates are non-increasing by construction. This chapter, and Lab 5.2, build the period-active table; the activity rule that defines “active in this period” is a measurement decision owing a data dictionary entry per Section 3.8, and naming the structure is part of that entry. Table 5.7 shows the period-active form with illustrative values.

Reading it is a learned skill worth two minutes. Reading along a row shows one cohort aging — period-active shares usually decline with age, though reactivation can lift a later month above an earlier one, which is a finding rather than an error. Reading down a column compares cohorts at the same age: a more comparable, same-age description, subject to the period and compositional differences that alignment does not remove, and exactly the “same-age comparison” the Chapter 2 memo committed to. Reading down the left edge watches cohort sizes, because a triumphant retention story over shrinking cohorts is two findings, not one. The lower-right of the table is empty by construction — young cohorts have not lived long enough to fill it — and the empty triangle is a feature: it displays, rather than hides, which comparisons the data can support.

Table 5.7

A period-active cohort retention table (illustrative values; share of cohort active in month since first purchase)

First-purchase cohort	Size	Month 0	Month 1	Month 2	Month 3
2026 Q1	620	100%	34%	27%	24%
2026 Q2	585	100%	32%	26%	—
2026 Q3	710	100%	30%	—	—
2026 Q4	655	100%	—	—	—

The funnel manages a different composition — the ordered stages of a journey. Descriptively, a funnel is a sequence of stage populations with counts at each stage and rates between them: impressions, clicks, sessions, carts, orders in an acquisition funnel; sent, delivered, opened, clicked in an email funnel. Its intellectual ancestry is the hierarchy-of-effects tradition in advertising — the idea that customers pass through ordered stages from attention to action (Strong, 1925; Lavidge & Steiner, 1961) — but this chapter needs it only as descriptive structure.

DEFINITION

Conversion Funnel

A conversion funnel is a descriptive structure that arranges a customer journey as an ordered sequence of stages, records the count of units reaching each stage, and reports conversion as rates between stages. Each stage's membership rule is an operational definition (Section 3.2) that must be stated, and every funnel rate must declare whether it is step-to-step (this stage over the previous stage) or cumulative (this stage over the top of the funnel).

A funnel's stages are strictly nested — each stage a subset of the one before — only when every stage counts the same unit, such as unique users or sessions. A sequence that mixes units, as impressions, clicks, and orders do, is a stage sequence rather than a set of nested populations, and must be labeled as such.

Source: Adapted from Strong (1925) and Lavidge and Steiner (1961).

Three disciplines make funnels trustworthy. The first is the definition discipline: stage boundaries are measurement decisions — what counts as a session, a cart, an “opened” email — and moving a boundary moves every rate downstream of it, which is why contested funnels are almost always definition disputes rather than behavior changes (a pattern Section 5.13 revisits in industry). The second is the denominator discipline in its funnel form: a “checkout conversion rate of 60 percent” (step) and a “visit-to-order conversion of 3 percent” (cumulative) can describe the same funnel, and a reader told one while imagining the other is misinformed by true numbers.

The third is the unit discipline, and it is the one most often skipped. Impressions and clicks are event counts, orders are transaction counts, one person generates many of the first two, and an attributed order may be credited on a view rather than a click. “Impressions → clicks → attributed orders” is therefore a campaign-performance sequence, not a nested population, and the arithmetic that nesting licenses — stage counts must be non-increasing, cumulative rates are products of step rates — does not apply to it. Two designs are defensible. Build the funnel on one consistent unit (unique users or sessions, counting click-attributed orders only) and verify that each stage's count does not exceed the previous stage's; or keep the event metrics separate, report click-through rate, click-to-attributed-order rate, and cost per attributed order as what they are, and state the attribution caveat in a sentence. Lab 5.2 takes the second route, because StyleCraft's campaign data carries impressions, clicks, and spend but no session-level events.

Both cohort tables and funnels, finally, submit to the chapter's verification theme: a cohort row's cells are computed over its stated size, and stage counts must reconcile to the certified tables they came from. Where a funnel is genuinely nested, its stage counts must also be non-increasing. Checking that attributed orders do not exceed total

orders is a necessary reconciliation, not a proof of nesting. Description along time and along the journey established, one section of context-setting remains before the toolkit goes to work.

5.9 Concentration, Benchmarks, and the Insight Statement

The toolkit can now count, center, spread, cross-tabulate, profile, decompose, and follow cohorts and funnels. This closing conceptual section adds the three finishing disciplines that turn competent description into usable insight: measuring concentration, supplying context, and writing the sentence.

Concentration first. Marketing quantities are not just skewed; they are concentrated — a minority of customers typically accounts for a majority of revenue, a pattern documented in consumer research for six decades as the "heavy half" (Twedt, 1964) and studied formally since (Schmittlein et al., 1993). The descriptive instrument is simple: rank customers by revenue over a stated window, cut the ranking into deciles (ten equal-count groups), and report each decile's share of total revenue. The headline is usually the top decile's share, the popular shorthand is the Pareto ratio ("the top 20 percent account for X percent"), and the working caution comes from the research: the true split is rarely the folkloric 80/20, it varies meaningfully by category and window, and measured concentration is inflated by observation error over short windows — so the window belongs in the sentence (Schmittlein et al., 1993).

DEFINITION

Concentration Analysis

Concentration analysis describes how unevenly a total — revenue, orders, engagement — is distributed across the units that produce it, typically by ranking units, cutting them into deciles or percentiles, and reporting each group's share of the total. Its summary claims (top-decile share; the "heavy half"; Pareto ratios) are window-dependent and must state the ranking variable, the window, and the population ranked.

Source: Adapted from Twedt (1964) and Schmittlein et al. (1993).

Concentration matters to the expansion review for a specific reason: it converts the heterogeneity finding of Section 5.4 into money. If the top decile of customers carries a large share of revenue, then the question "which customers does the suburban expansion acquire?" is not answered by average anything — it is answered by where those customers land in the concentration ranking, and by whether their weak repeat behavior keeps them there. A decile table is one groupby away and reframes the entire meeting.

Context second. A number in isolation is not yet a claim; a number becomes a claim when placed against a benchmark — a reference value that supplies the comparison the reader will otherwise invent. Benchmarks come in three practical grades. Internal-historical (this quarter against last quarter, this cohort against the prior cohort at the same age) is the most trustworthy, because definitions and measurement are held constant. Internal-cross-sectional (this store against sister stores, this campaign against the campaign portfolio) is nearly as strong, subject to the composition cautions of Section 5.5. External (industry reports, published category rates) is the weakest and the most seductive: external benchmarks almost never share your metric's four elements — numerator, denominator, window, filters (Section 3.5.1) — so a gap against them is as often a definition gap as a performance gap, and the honest use is directional context, clearly labeled. Benchmarking is this chapter's descriptive cousin of the baseline discipline Chapter 2 introduced for interpretation (Section 2.7): both exist because a number's meaning is a comparison, and the analyst's job is to make the comparison explicit and fair rather than leave it to imagination.

Third, the sentence. The difference between a competent analyst and a trusted one is frequently one sentence long: the difference between an observation and an insight statement. An observation states what the data shows: "suburban AOV is 2.5 times urban AOV." An insight statement adds the three things a decision-maker needs and description can honestly supply: the decomposition or comparison that explains the number's composition, the magnitude in decision-relevant units, and the implication for the named decision — while stopping, deliberately, short of causal language the description cannot support. "Suburban orders are 2.5 times urban orders in value,

factoring into a 1.67× basket-size factor and a 1.50× price-per-unit factor that tracks occasionwear mix rather than list prices — but suburban customers order far less often, so their revenue per customer leads by much less than the AOV headline suggests, and the expansion case should be argued on revenue per customer, not AOV" is an insight statement. It is longer than the observation, every clause is checkable against a table in this chapter, and nothing in it claims to know why.

> **CONCEPT**
>
> **Insight Statements versus Observations**
>
> An observation reports a number; an insight statement equips a decision. An insight statement (1) names the comparison or decomposition that gives the number meaning, (2) states magnitude in the decision's units, (3) connects to the named decision from the analytic specification (Section 2.5), and (4) stays inside descriptive language — "is associated with," "accounts for," "coincides with" — leaving causal verbs to the evidence that can carry them (Chapters 7 and 11 build that evidence). If a claim's verb asserts more than the tables show, it is not insight; it is speculation with a citation to itself.

With the conceptual sequence complete — precise counting, honest centers and spreads, composition-aware comparison, the metric families and the decomposition identity, cohorts, funnels, concentration, benchmarks, and the insight sentence — the chapter turns to the collaborator who will draft most of this work's code in practice, and to the audit that collaboration requires.

5.10 AI as a Descriptive Analytics Assistant

Descriptive analytics is the domain where AI assistance is at its most fluent and, for exactly that reason, at its most quietly dangerous. Every tool in this chapter — frequency tables, groupbys, cross-tabs, pivot tables, cohort matrices, decile cuts — is one short prompt away from working pandas code, and the code will very likely be correct. The risk has moved: in descriptive work, assistants rarely fail at computation; they fail at the analytic decisions hiding inside the computation, and they fail silently, because the output is a tidy table that looks finished.

Where the assistant genuinely helps, use it. Drafting the mechanical layer — groupby-aggregate scaffolds (whose mechanics Chapter 4's lab established), crosstab and pivot calls, cohort-table reshaping, formatting for presentation — is legitimate delegation that compresses hours into minutes. Asking an assistant to propose which descriptive cuts might illuminate a question is a useful brainstorm, provided proposals are treated as candidates for the specification, not as analysis. And a first-pass narration of a finished, verified table can accelerate writing, provided every sentence survives the insight-statement discipline of Section 5.9.

What must never be delegated are the decisions this chapter spent nine sections on, because they are judgment calls wearing default settings. The characteristic failure modes are worth naming individually. The silent mean: asked to "summarize order value," assistants overwhelmingly reach for the mean — on data Section 4.9.1 already showed is skewed — and report a center the median contradicts. The invented denominator: asked for a "repeat rate" or "open rate," the assistant picks a denominator, does not say which, and the four-element discipline of Section 3.5.1 dies in a code comment. The averaged average: asked to summarize by group and then overall, assistants routinely average the group means — the exact trap of Section 5.3 — producing an overall figure that reconciles to nothing. The pooled comparison: asked to compare groups, assistants compare aggregates with no composition check, walking directly into Section 5.5's paradox. The unreconciled table: filtered summaries whose totals no longer match the certified file, with the filter mentioned nowhere. And the narrative overreach: asked to "explain" a table, assistants supply causal stories — customers "because," stores "driving" — converting description into speculation at exactly the step stakeholders will quote.

AI IN PRACTICE

The Summary That Must Add Up

A productive pattern for delegated description: paste the relevant data dictionary entries and the certified totals from the Chapter 4 verification log (row count, distinct orders and customers, total revenue — never confidential data; the synthetic StyleCraft files are safe), then prompt — "Write pandas code to produce [one specific table]. Before the code, state: the unit of analysis of each output row; the exact denominator of every rate or proportion; which measure of center you will use and why, given that order value is right-skewed; and the reconciliation I should expect — which totals of your table must equal the certified totals above. After the code, print those reconciliation figures." Then audit in three moves: recompute one cell by hand from the certified table; check the printed totals against the log; and strike every causal verb from any narration before it travels. The predictions come first, per Section 1.7's predict-then-verify, and the whole exchange is documented per Appendix D — the assistant drafted the table, but the analyst of record certified that it adds up.

The deeper point extends this guide's standing division of labor into new territory. In Chapter 4 the assistant drafted cleaning code and the analyst audited the counts; here the assistant drafts summaries and the analyst audits the claims — denominators, centers, compositions, reconciliations, and verbs. Appendix C provides prompt templates for the common descriptive tasks; Appendix D's documentation requirement applies to every exchange; and the labs that follow are built so that the audit has teeth, because on StyleCraft's designed data the right answers are known.

5.11 Hands-On Application in Python and Google Colab

The preceding sections built the toolkit conceptually. This section runs it against the opening case. Lab 5.1 performs the chapter's central move — decomposing the suburban-versus-urban average-order-value gap — twice: first on an eight-order miniature constructed in the notebook, so every figure can be verified by hand, and then on StyleCraft's full certified data, where the decomposition recovers a finding the dataset was designed to contain. Lab 5.2 builds the review's remaining descriptive artifacts: profiles, a cohort retention table, a concentration analysis, and campaign funnel metrics. Throughout, follow the course division of labor: AI assistants may draft the code (Appendix C has prompting templates; Appendix A covers Colab mechanics), but every summary is predicted before it is computed, reconciled to the certified totals afterward, and every AI exchange is documented per Appendix D.

5.11.1 Lab 5.1, Part A: The Decomposition by Hand

The miniature below contains eight orders — six urban, two suburban — as fourteen order lines, with quantities and prices inside the catalog's $24-to-$90 band and one column, is_occasionwear, carried in from the products dimension. It is built so that the pooled and group summaries are hand-checkable and so that the decomposition identity of Section 5.7 can be verified with a calculator. Before running the first cell, start the predictions: given six urban orders and two suburban ones, will the simple average of the two metro AOVs equal the pooled AOV? (Section 5.3 says it will not, and says which one reconciles to total revenue.)

Code 5.1. Create the hand-checkable miniature

```
import numpy as np
import pandas as pd

lines = pd.DataFrame({
    "order_id":  ["O5001","O5002","O5002","O5003","O5004","O5004",
                  "O5005","O5006","O5006",
                  "O5007","O5007","O5007","O5008","O5008"],
    "metro":     ["NYC Urban"]*9 + ["NYC Suburban"]*5,
```

```
    "quantity":  [1]*14,
    "unit_price":[34.0, 36.0, 30.0, 38.0, 28.0, 40.0,
                  80.0, 38.0, 36.0,
                  72.0, 78.0, 36.0, 75.0, 39.0],
    "is_occasionwear": [False, False, False, False, False, False,
                        True,  False, False,
                        True,  True,  False, True,  False]
})
lines["line_revenue"] = lines["quantity"] * lines["unit_price"]

# Part A's certified totals, written as checks rather than as prose.
assert len(lines) == 14
assert lines["order_id"].nunique() == 8
assert np.isclose(lines["line_revenue"].sum(), 660.0)

# metro is carried through the grain change below, so it must be
# constant within every order.
assert (lines.groupby("order_id")["metro"]
             .nunique(dropna=False).le(1).all())

print("lines:", len(lines),
      "| orders:", lines["order_id"].nunique(),
      "| revenue:", lines["line_revenue"].sum())
```

Expected output: 14 lines, 8 orders, revenue 660.0, and four assertions that pass silently.

Input: fourteen literal order lines. Transformation: one derived column, line_revenue. Output: the three certified totals every later summary must reconcile to, in the spirit of Chapter 4's verification log. The assertions are the point of the cell. Chapter 4 argued that a prediction belongs in the log before the step runs; an assertion is that prediction written where the notebook itself will enforce it, and a printed figure a student forgets to read is not a check. The fourth assertion guards the grain change that follows: carrying metro from lines to orders is honest only if every line in an order shares one metro.

The first computation changes grain deliberately (Section 3.3; mechanics per Section 4.8): order value lives at the order grain, so lines are grouped to orders before any AOV is touched. Predict the output shape first: eight rows, one per order.

Code 5.2. Change grain from lines to orders, then AOV by metro

```
orders = (lines
          .groupby(["order_id", "metro"], as_index=False)
          .agg(order_value=("line_revenue", "sum"),
               units=("quantity", "sum")))

# The grain change must preserve the certified totals exactly.
assert len(orders) == lines["order_id"].nunique()
assert np.isclose(orders["order_value"].sum(),
                  lines["line_revenue"].sum())

aov = (orders
       .groupby("metro")
       .agg(orders=("order_id", "count"),
            revenue=("order_value", "sum"),
            aov=("order_value", "mean"),
            units_per_order=("units", "mean")))

pooled_aov   = orders["order_value"].sum() / len(orders)
weighted_aov = ((aov["aov"] * aov["orders"]).sum()
                / aov["orders"].sum())
simple_aov   = aov["aov"].mean()
```

```
# Weighted recombination reconciles to the pooled figure;
# the simple average of group means does not.
assert np.isclose(pooled_aov, weighted_aov)

print(orders)
print(aov)
print("pooled:", round(pooled_aov, 2),
      "| weighted:", round(weighted_aov, 2),
      "| simple average of metro AOVs:", round(simple_aov, 2))
```

Expected output: an eight-row orders frame; urban AOV 60.00 over 6 orders and suburban 150.00 over 2; pooled 82.50, weighted 82.50, and the simple average of the metro AOVs at 105.00.

Input: the fourteen certified lines. Transformation: rows are collected by order (and by the metro that travels with the order), then summarized by metro. Output: the headline gap, plus the averaged-average trap made numerical. The two assertions state the grain-change identities of Section 4.8 — one row per distinct order, and a preserved total — and the third states Section 5.3's standing rule: the weighted recombination equals the pooled figure, while the $105.00 simple average reconciles to no total in the file.

VERIFICATION CHECK

Before you run: write four predictions before running. (1) Reconciliation: the two metros' order counts must sum to 8 and their revenues to 660.0 — the known-totals check of Section 5.7. (2) Urban AOV: urban revenue is 360.0 over 6 orders, so 60.0; suburban is 300.0 over 2 orders, so 150.0. (3) The pooled AOV is 660/8 = 82.50; the simple average of the metro AOVs is (60 + 150)/2 = 105.00 — the averaged-average gap of Section 5.3 — and the weighted recombination (6×60 + 2×150)/8 must return exactly 82.50. (4) Units per order: urban 9/6 = 1.5; suburban 5/2 = 2.5.

After you run: every assertion passes silently, the printed metro table matches predictions (2) and (4), and the three AOV figures read 82.50, 82.50, and 105.00.

Investigate if: any assertion raises. A failed row-count assertion means the grain change dropped or duplicated something; a failed total assertion means a group key contained a missing value; a failed weighted-recombination assertion means the weights are wrong. Find the cause before proceeding.

With the headline gap on screen — suburban AOV of 150 against urban 60, a ratio of 2.5 — the decomposition identity does its work. AOV = units per order × revenue per unit, so the 2.5× gap must factor exactly into a basket factor and a price-per-unit factor.

Code 5.3. Decompose AOV into basket size and revenue per unit

```
decomp = aov.copy()
decomp["units"] = decomp["units_per_order"] * decomp["orders"]
decomp["revenue_per_unit"] = decomp["revenue"] / decomp["units"]
decomp["identity_check"] = (decomp["units_per_order"]
                            * decomp["revenue_per_unit"])

# Verification 1: the factors multiply back in every group.
assert np.allclose(decomp["identity_check"], decomp["aov"])

sub, urb = "NYC Suburban", "NYC Urban"
ratio_aov    = decomp.loc[sub, "aov"] / decomp.loc[urb, "aov"]
ratio_basket = (decomp.loc[sub, "units_per_order"]
                / decomp.loc[urb, "units_per_order"])
ratio_price  = (decomp.loc[sub, "revenue_per_unit"]
                / decomp.loc[urb, "revenue_per_unit"])

# Verification 2: the ratio of composites is the product of
```

```
# the ratios of the factors.
assert np.isclose(ratio_aov, ratio_basket * ratio_price)

print(decomp[["aov", "units_per_order",
              "revenue_per_unit", "identity_check"]])
print(round(ratio_aov, 4), "=",
      round(ratio_basket, 4), "x", round(ratio_price, 4))
```

Expected output: revenue per unit of 40.0 urban and 60.0 suburban; identity_check equal to aov in both rows; and the ratio line reading 2.5 = 1.6667 × 1.5.

Input: the metro summary. Transformation: the composite is written as the product of its two factors, group by group. Output: the decomposition and its two verifications. Read the result in the vocabulary Section 5.7 insisted on: the 2.5× AOV ratio *factors* into a 1.67× basket-size factor and a 1.50× realized-price-per-unit factor. It does not divide the gap into percentage shares, and a sentence claiming that baskets explain "two-thirds of the gap" would be asserting an allocation this identity does not produce.

The last step interrogates the price factor. List prices for the same product do not differ by store, although realized price per unit can still differ because of discounts, product mix, or promotion mix — so a 1.5× gap in revenue per unit is a claim about what was bought, not about what things cost, and the miniature carries the category flag to examine it.

Code 5.4. Examine category mix and realized prices

```
mix = (lines
       .groupby(["metro", "is_occasionwear"])
       .agg(units=("quantity", "sum"),
            revenue=("line_revenue", "sum")))
mix["avg_price"] = mix["revenue"] / mix["units"]

# Occasionwear unit share, computed with explicit columns rather
# than groupby.apply, whose treatment of the grouping columns is
# both version-sensitive and hard for a reader to audit.
share = (lines
         .assign(occasionwear_units=lines["quantity"]
                 .where(lines["is_occasionwear"], 0))
         .groupby("metro")
         .agg(occasionwear_units=("occasionwear_units", "sum"),
              total_units=("quantity", "sum")))
share["occasionwear_unit_share"] = (share["occasionwear_units"]
                                    / share["total_units"])

# Standardized comparison: each metro's own mix, priced at the
# urban metro's within-category average prices.
urban_price = mix.loc["NYC Urban", "avg_price"]
share["revenue_per_unit_at_urban_prices"] = (
    share["occasionwear_unit_share"] * urban_price[True]
    + (1 - share["occasionwear_unit_share"]) * urban_price[False]
)

print(mix.round(2))
print(share[["occasionwear_unit_share",
             "revenue_per_unit_at_urban_prices"]].round(3))
```

Expected output: within-category average prices of $80.00 (occasionwear) and $35.00 (other) in the urban metro against $75.00 and $37.50 in the suburban metro; occasionwear unit shares of 0.111 and 0.600; and standardized revenue per unit of $40.00 urban and $62.00 suburban.

Input: the fourteen lines with their category flag. Transformation: a two-way summary of units, revenue, and realized price; an explicit occasionwear unit share; and one standardization that holds within-category prices constant while letting the mix vary. Output: three views that together locate the price factor.

Read them in order. The metros do not pay materially different prices for the same kind of item — the within-category averages sit a few dollars apart, and in opposite directions — while the occasionwear share of units differs by more than five times. The standardization makes the comparison exact: buying its own mix at the urban metro's within-category prices, the suburban metro would post revenue per unit of $62.00 against the urban $40.00, a factor of 1.55, slightly *more* than the 1.50 factor actually observed. Mix, in other words, more than accounts for the price factor, and the small within-category price differences work marginally against it.

Be precise about what this establishes. The miniature shows a revenue-per-unit gap strongly associated with category mix, together with small within-category differences in realized price. It does not show that identical products sold at identical realized prices, which would require product identifiers the miniature does not carry — and which Part B, working on the full file with product_id available, can check directly. Section 5.7's identity is exact by construction; this second step is a standardization, and keeping the difference straight is the difference between a decomposition and an explanation.

VERIFICATION CHECK

Before you run: predict that urban units split 1 occasionwear to 8 other and suburban 3 to 2, giving occasionwear unit shares of 0.111 and 0.600. Predict that within-category average prices will be close across metros but not identical. Then predict the recombination: each metro's (occasionwear share × its own occasionwear price + other share × its own other price) must reproduce its revenue per unit — 40.0 and 60.0 — exactly.

After you run: the shares are 0.111 and 0.600; urban within-category prices are $80.00 and $35.00 against suburban $75.00 and $37.50; each metro's recombination returns 40.0 and 60.0; and the standardized figures are $40.00 and $62.00.

Investigate if: a recombination misses its metro's revenue per unit. That means a share or a price was computed over the wrong base: both are unit-weighted, so units — not order lines and not orders — is the denominator for each.

Close Part A the way Chapter 4 closed its miniature: with the certified account. Eight orders and 660.0 in revenue, fully reconciled; a 2.5× AOV gap factored exactly into a 1.67× basket factor and a 1.50× realized-price-per-unit factor; and the price factor shown to track category mix — 60 percent of suburban units are occasionwear against 11 percent of urban, and at common within-category prices that mix alone would produce a 1.55× factor. Every number above is reproducible with a calculator from the fourteen lines, and reproducing them is the point: the full file will not be hand-checkable, so the habits must be certified while it is.

5.11.2 Lab 5.1, Part B: The Designed Finding at Full Scale

Part B runs the same sequence on StyleCraft's certified data — transactions_clean (or your own certified equivalent from Lab 4.1), joined to products for is_occasionwear and to stores for metro, all available from Brightspace and the companion repository. The dataset was designed so that the suburban and resort AOV advantage is produced by basket size and occasionwear mix, not unit price; your decomposition either recovers that designed finding or it does not, and either outcome is gradable — this is the book's known-truth verification thread at work, the same arrangement that gave Chapter 4's lab an answer key.

Work in the Part A order, with five scale disciplines added. First, open the log: record the certified totals (rows, distinct orders, distinct customers, total revenue) from your Chapter 4 verification log before computing anything, and reconcile every groupby against them. Second, predict directions before magnitudes: before the AOV cell runs, write down which metro group will lead, whether the basket factor or the price factor will contribute more, and roughly where the occasionwear unit shares will land — then compare. Third, use the product identifiers the

miniature lacked: for the products that sell in both metros, compare realized price per unit directly, which turns Part A's standardized argument into a direct test of whether the same item transacts at the same price in both places. Fourth, let an AI assistant draft each step per Section 5.10's AI in Practice pattern, requiring its stated denominators and reconciliation figures in advance; at full scale at least one delegated summary will fail a reconciliation or default to the wrong center, and finding it is part of the assignment. Fifth, close with the identity audit: factors multiply back in every group, weighted recombination reproduces the pooled AOV, and within-category average prices sit inside the catalog band in every metro.

> **VERIFICATION CHECK**
>
> Before you run: predict the metro AOV table, the two factors, and the occasionwear unit shares before any cell runs, and write down which of the two factors you expect to be larger.
>
> After you run: the factors multiply back in every metro, the weighted recombination reproduces the pooled AOV, and the same-product realized-price comparison shows no systematic metro difference.
>
> Investigate if: your decomposition table differs from a classmate's in any way not attributable to a stated choice. The designed finding is the answer key, not the assignment: matching it does not excuse a missing reconciliation, and a defensible pipeline that reveals the same structure with slightly different figures (from different but documented join or filter choices) is better work than matching numbers with an unlogged filter. An unexplained difference means a silent step ran somewhere — the Chapter 4 rule, now applied to summaries.

5.11.3 Lab 5.2: Profiles, Cohorts, Concentration, and Campaign Metrics

Lab 5.2 builds the review's remaining artifacts on the certified data, one per part, each opened with a written prediction and closed with a reconciliation. Each part below ships a worked cell that demonstrates the pattern and leaves a stated extension for homework; the cells assume transactions_clean, customers, and campaigns are loaded, that order_date has been parsed to a true date, and that numpy and pandas are imported as in Code 5.1.

Part 1, the profile. Complete Table 5.5: build the customer-grain columns by grouping transactions to customers, the order-grain columns from the Part B orders table, and the line-grain occasionwear share; report mean and median together wherever Section 5.3's skew rule applies, group sizes first per Section 5.6. Code 5.5 builds the customer-grain half of the profile and its reconciliations; the order-grain and line-grain rows are yours to add.

Code 5.5. Build the customer-grain profile

```
# Step 1: transactions to orders, then orders to customers.
orders = (transactions_clean
          .groupby(["order_id", "customer_id"], as_index=False)
          .agg(order_value=("line_revenue", "sum"),
               units=("quantity", "sum"),
               order_date=("order_date", "min")))

assert len(orders) == transactions_clean["order_id"].nunique()
assert np.isclose(orders["order_value"].sum(),
                  transactions_clean["line_revenue"].sum())

cust = (orders
        .groupby("customer_id", as_index=False)
        .agg(orders=("order_id", "count"),
             revenue=("order_value", "sum"),
             units=("units", "sum")))

# The headline repeat-purchase definition of Section 3.10.3:
# two or more orders in the observation window.
cust["is_repeater"] = cust["orders"] >= 2
```

```
# Step 2: attach the grouping dimension, cardinality declared.
cust = cust.merge(customers[["customer_id", "home_metro"]],
                  on="customer_id", how="left",
                  validate="many_to_one")
print("customers with no home metro:",
      int(cust["home_metro"].isna().sum()))

# Step 3: the profile. dropna=False keeps any unmatched customers
# visible instead of deleting them from the denominator.
profile = (cust
           .groupby("home_metro", dropna=False)
           .agg(customers=("customer_id", "count"),
                orders_mean=("orders", "mean"),
                orders_median=("orders", "median"),
                repeat_rate=("is_repeater", "mean"),
                revenue_mean=("revenue", "mean"),
                revenue_median=("revenue", "median")))

# Reconciliations: the profile counts every customer and every dollar.
assert profile["customers"].sum() == len(cust)
assert np.isclose((profile["customers"] * profile["revenue_mean"]).sum(),
                  transactions_clean["line_revenue"].sum())

print(profile.round(2))
```

Expected output: a printed count of customers with no home metro (zero on a correctly joined certified file), one profile row per metro, all four assertions passing, and a mean above the median in every revenue column — the skew of Section 4.9.1 arriving at customer grain.

Input: the certified line-grain table and the customers dimension. Transformation: two deliberate grain changes (lines to orders, orders to customers) and one validated join. Output: the customer-grain half of Table 5.5, with group sizes first. The second assertion is the chapter's known-totals check in profile form: the customer counts times the group revenue means recombine — weighted, never simply averaged — to the certified total. Extension for homework: add the order-grain rows (AOV, units per order) from the Part B orders table and the line-grain occasionwear share, and label each row with its grain as Table 5.5 requires.

Part 2, the cohort table. Define first-purchase monthly cohorts, choose and document an activity rule (the dictionary entry is part of the deliverable, per Section 3.8, and it must name the table's structure), and produce the retention table split urban versus suburban. Read it down the columns: at the same age, how do the metros compare — and does the pooled repeat-rate gap survive the same-age comparison? This part is the same-age cohort analysis the Chapter 2 memo committed to, and its verification is structural: each row's cells are proportions of that row's stated size, and the young-cohort triangle stays empty.

Code 5.6. Construct the period-active cohort retention table

```
# Activity rule (dictionary entry): a customer is ACTIVE in a
# period if the customer placed at least one order in that
# calendar month. This is a PERIOD-ACTIVE table, per Section 5.8,
# so a cell may exceed the cell to its left when customers return.
orders["order_month"] = orders["order_date"].dt.to_period("M")

cohort = (orders.groupby("customer_id")["order_month"]
          .min().rename("cohort"))
orders = orders.join(cohort, on="customer_id")
orders["period"] = [(m - c).n for m, c
                    in zip(orders["order_month"], orders["cohort"])]

size = cohort.value_counts().sort_index().rename("size")
```

```
# A complete, contiguous period axis. Without the reindex, a period
# in which no cohort happened to be active would drop out of the
# table altogether and the ages would silently stop being adjacent.
periods = pd.Index(range(int(orders["period"].max()) + 1),
                   name="period")

# Two different absences have to be filled here, and reindex alone
# fills only one of them. unstack leaves NaN for a (cohort, period)
# pair that produced no rows; reindex adds whole periods that no
# cohort reached. fillna after the reindex covers both, so every
# cell is a count before any masking happens.
active = (orders
          .drop_duplicates(["customer_id", "period"])
          .groupby(["cohort", "period"])["customer_id"]
          .count()
          .unstack("period")
          .reindex(index=size.index, columns=periods)
          .fillna(0))

# Zero activity and an unobserved future are different facts and
# must not share a cell value. Elapsed-but-inactive cells are 0.0;
# periods a cohort has not yet lived through are masked to missing.
analysis_month = orders["order_month"].max()

observable = pd.DataFrame(
    [[(cohort_month + p) <= analysis_month for p in periods]
     for cohort_month in size.index],
    index=size.index,
    columns=periods,
)

retention = active.div(size, axis=0).where(observable)

# Structural verifications.
assert (retention[0] == 1.0).all()          # period 0 is the cohort itself
assert size.sum() == orders["customer_id"].nunique()
assert retention.max().max() <= 1.0
assert retention.isna().equals(~observable)  # missing means unobserved

print(size)
print((retention.iloc[:, :7] * 100).round(1))
```

Expected output: a cohort-size column summing to the distinct customer count; a retention table with contiguous period columns whose first column is 100 percent in every row; 0.0 wherever a cohort lived through a period with nobody active; and blanks only in the lower-right triangle no cohort has yet reached.

Input: the order-grain table from Code 5.5. Transformation: each customer is assigned to the month of the first order; each order is placed at its age in months since that cohort month; the count of distinct customers active at each age is divided by the cohort's size; and the result is masked to the periods that have actually elapsed. Output: the period-active table of Table 5.7. The reindex and the mask together carry the section's most easily lost distinction. A blank cell in a cohort table can mean two opposite things — the cohort reached that age and nobody bought, or the cohort has not reached that age at all — and a table that renders both as missing invites the reader to mistake a collapse for an absence of data. Filling elapsed cells with zero and masking only the unobserved future makes the empty triangle mean exactly one thing, which is what licenses reading down a column. The four assertions encode what the structure guarantees: everyone is active at period 0, the cohort sizes partition the customer base, no cell can exceed 100 percent, and a cell is missing if and only if the period has not yet occurred. Note what is deliberately *not* asserted — that each row declines — because in a period-active table a return visit legitimately raises a later month. Extension for homework: split the table by home_metro and read the two versions down the columns.

Part 3, concentration. Rank customers by revenue over the trailing twelve months, cut deciles, and report each decile's share of revenue plus the top-decile and top-two-decile shares, window stated per Section 5.9. Verification: decile revenues must sum to the window's total revenue exactly. Then the review question: cross the top revenue decile against home metro — where do the expansion's customers land?

Code 5.7. Build revenue deciles and concentration shares

```
# The window is part of the claim, so it is declared, not implied.
window_end   = transactions_clean["order_date"].max()
window_start = window_end - pd.DateOffset(months=12)
window = transactions_clean[
    transactions_clean["order_date"] > window_start
]

# Every dollar ranked must belong to a customer, or the deciles
# and the window total will not reconcile.
assert window["customer_id"].notna().all(), \
    "lines without a customer cannot be ranked; resolve them first"

rev = (window
       .groupby("customer_id", as_index=False)
       .agg(revenue=("line_revenue", "sum")))

# Decile 1 is the highest-revenue tenth. rank(method="first")
# breaks ties deterministically so the cut is reproducible.
rev["decile"] = pd.qcut(
    rev["revenue"].rank(method="first", ascending=False),
    10, labels=range(1, 11)
)

conc = (rev.groupby("decile", observed=True)
        .agg(customers=("customer_id", "count"),
             revenue=("revenue", "sum")))
conc["share_of_revenue"] = conc["revenue"] / conc["revenue"].sum()

# The deciles must partition the window exactly.
assert conc["customers"].sum() == len(rev)
assert np.isclose(conc["revenue"].sum(), window["line_revenue"].sum())

print(f"window: after {window_start.date()} "
      f"through {window_end.date()}")
print(conc.round(3))
print("top decile share:",
      round(float(conc["share_of_revenue"].iloc[0]), 3))
print("top two deciles:",
      round(float(conc["share_of_revenue"].iloc[:2].sum()), 3))
```

Expected output: the window printed as the interval it actually is — open at the start, closed at the end — then ten decile rows of roughly equal customer counts with steeply unequal revenue shares, all three assertions passing, and top-decile and top-two-decile shares well above ten and twenty percent respectively.

Input: the certified line-grain table. Transformation: a declared twelve-month window, a customer-grain revenue total inside it — the filter is strictly greater than the start date and inclusive of the end date, so the interval is (window_start, window_end] and the printed label says so — a ranked cut into ten equal-count groups, and each group's share of the window's revenue. Output: the concentration table and the two headline shares. The assertions carry the section's caution: deciles are a partition, so the customer counts and the revenue must both add back to the window, and any figure quoted from this table travels with the window that produced it (Schmittlein et al., 1993). Extension for homework: cross the top decile against home_metro and against loyalty tier, and repeat the whole cell on a ninety-day window to see how much the headline moves.

Part 4, the campaign metrics. From the campaigns table, compute CTR (per Section 1.10's construction) and CPC per campaign and per channel, labeling every rate step-versus-cumulative and naming denominators; then extend to the campaign-performance sequence of Section 5.8 — impressions, clicks, attributed orders — reporting the click-to-attributed-order rate and cost per attributed order, and noting in one sentence that these are three different units of count rather than a nested funnel. Attribution is itself a definition: campaign_id's nulls carry the two-kinds-of-missing ambiguity documented in Chapter 4.

Code 5.8. Calculate campaign and funnel metrics

```
# The fact table is at line grain, so one order can carry several
# campaign_id values unless the attribution rule forbids it. If it
# can, distinct orders summed across campaigns double-count.
assert (transactions_clean
        .groupby("order_id")["campaign_id"]
        .nunique(dropna=True)
        .le(1)
        .all()), "at least one order carries multiple campaign identifiers"

attributed = (transactions_clean
              .dropna(subset=["campaign_id"])
              .groupby("campaign_id", as_index=False)
              .agg(attributed_orders=("order_id", "nunique"),
                   attributed_revenue=("line_revenue", "sum")))

perf = campaigns.merge(attributed, on="campaign_id",
                       how="left", validate="one_to_one")
perf[["attributed_orders", "attributed_revenue"]] = (
    perf[["attributed_orders", "attributed_revenue"]].fillna(0)
)

def add_rates(frame):
    """Four rates, four denominators, always recomputed from counts.
    A zero denominator becomes missing rather than infinite, so a
    campaign with no clicks reports no rate instead of a record one."""
    imp  = frame["impressions"].replace(0, np.nan)
    clk  = frame["clicks"].replace(0, np.nan)
    ordr = frame["attributed_orders"].replace(0, np.nan)
    frame["ctr"] = frame["clicks"] / imp
    frame["cpc"] = frame["spend"] / clk
    frame["orders_per_click"] = frame["attributed_orders"] / clk
    frame["cost_per_attributed_order"] = frame["spend"] / ordr
    return frame

perf = add_rates(perf)

# Channel roll-up: pool the counts, then run the same function again.
# Rates are recomputed, never averaged across campaigns (Section 5.3).
by_channel = add_rates(
    perf.groupby("channel").agg(
        impressions=("impressions", "sum"),
        clicks=("clicks", "sum"),
        spend=("spend", "sum"),
        attributed_orders=("attributed_orders", "sum")))

# Reconciliations. These check the sequence against its sources;
# they do NOT establish that the stages are nested populations.
assert (perf["attributed_orders"].sum()
        <= transactions_clean["order_id"].nunique())
assert np.isclose(perf["spend"].sum(), campaigns["spend"].sum())
assert (perf["ctr"].dropna() <= 1).all(), \
```

```
    "clicks exceed impressions somewhere"
print(perf[["campaign_id", "channel", "impressions", "clicks",
            "spend", "attributed_orders", "ctr", "cpc",
            "orders_per_click",
            "cost_per_attributed_order"]].round(4).head(10))
print(by_channel.round(4))
```

Expected output: a per-campaign table of impressions, clicks, spend, attributed orders and the four rates; a channel roll-up carrying the same four rates recomputed from pooled counts; four assertions passing; and a missing value, rather than an infinity, wherever a campaign or channel produced no clicks or no attributed orders.

Input: the campaigns dimension and the certified transactions. Transformation: attributed orders and revenue are rolled up per campaign, joined one-to-one to the campaign row, and converted into the same four rates at two levels by one shared function, with every zero denominator converted to a missing value first — a rate over nothing is undefined, not infinite, and a table full of inf is a denominator problem wearing a printing error. Output: campaign performance as a sequence of event metrics at two levels. The shared add_rates function is doing more than saving lines: because the channel figures are produced by pooling the counts and running the same function again, the weighted recombination of Section 5.3 is structural rather than advisory — there is no code path in this cell that could average the campaigns' rates, which is exactly the number that would reconcile to nothing. Note the first assertion too: because the fact table is at line grain, an order carrying two campaign identifiers would be counted once under each, and the total-orders reconciliation could still pass while the per-campaign figures double-count. Note finally what the assertions do not claim — attributed orders not exceeding total orders is a reconciliation, not a proof that orders nest inside clicks (Section 5.8). Extension for homework: add a cumulative rate from impressions to attributed orders, state in the notebook why it is not a conversion rate in the nested sense, and report how many attributed orders would be lost under a click-only attribution rule.

Deliverables for both labs: the notebook, the reconciliation log, the completed profile and cohort tables, the data dictionary entries for the activity rule and the attribution rule, and the AI-Use Appendix (Appendix D) documenting every assistant exchange, including at least one delegated summary you corrected and why.

5.12 Marketing Interpretation and Managerial Insight

The labs produced tables; the review needs sentences. This section returns to the meeting about one number and translates the chapter's outputs into decision language — including, per this guide's standing practice, the wrong reading that must be corrected before it circulates.

The headline the analytics lead can now deliver is a decomposition, not a defense or a debunking. The suburban AOV advantage is real, it reconciles to certified totals, and it is made of two measurable things: bigger baskets and a mix that skews to occasionwear at the top of an unchanged price band — with each factor stated exactly, courtesy of Section 5.7's identity. That single sentence converts the pre-read's Rorschach test into an agenda: if the advantage is baskets and mix, then the questions worth the room's time are whether occasion-driven baskets repeat (the cohort table answers at like ages), what the suburban customer is worth per customer rather than per order (the profile's revenue-per-customer row), and where those customers land in the concentration ranking that carries the company's revenue.

Now the wrong managerial reading, because it is already in circulation: “Suburban AOV is 2.5 times urban, so suburban customers are 2.5 times as valuable — resume the openings.” Every clause outruns the tables. AOV is an order-grain mean; customer value is a customer-grain quantity, and the bridge between them is order frequency — revenue per customer equals AOV times orders per customer — which is precisely where the suburban profile is weakest. A group can lead on AOV and trail on revenue per customer, and whether StyleCraft's suburban group does is an empirical cell in Table 5.5, not a matter of emphasis. Nor does even a favorable revenue-per-customer comparison carry the conclusion by itself: the pooled comparison mixes customer vintages (the new stores' customers are structurally young), which is why the memo demanded the same-age cohort comparison, and Section 5.5 showed with two thousand illustrative customers how badly a pooled rate can misrepresent every stratum it contains. The corrected sentence for the room is the insight statement built in Section 5.9 — AOV factored,

frequency named, the expansion case re-argued on revenue per customer at like ages — and its final clause matters most: nothing in this chapter says why any of it is so. “The suburban format attracts occasion shoppers” is a hypothesis the description makes plausible; testing whether format causes behavior is Chapter 11's kind of evidence.

One more translation earns its three minutes of meeting time: the definitions underneath the tables. The repeat rate uses the Chapter 3 headline definition; the cohort activity rule, the cohort table's period-active structure, and the attribution rule behind the campaign metrics are documented measurement choices; every rate's denominator is named in its table note. Surfacing this is not pedantry — it is the reason the meeting cannot dissolve into the “one metric, three numbers” dispute that consumed Chapter 3's version of this room, and it is what the analyst of record's signature on a descriptive deliverable means.

5.13 Business Analytics in Practice

This section turns from the fictional case to how the same descriptive disciplines appear in industry — where descriptive analytics is not the remedial tier of the field but its daily, decision-carrying core. The three vignettes below are composite: they synthesize recurring practices in retail, subscription, and digital analytics teams rather than reporting on any one named organization, and they are offered as patterns a new analyst will recognize rather than as prevalence claims about most companies.

5.13.1 The Monday Trading Report

The first vignette is the most ordinary artifact in retail analytics: the Monday trading report. In many retail and e-commerce organizations, the analytical product executives actually read every week is descriptive — a page or two, assembled by an analyst or an automated job overnight into Monday morning, summarizing the trading week. Its conventions are worth studying because they encode this chapter. The reading order is fixed and hierarchical: headline revenue against plan and against last year first (benchmarks before raw numbers, always — a number without its comparison is not yet information); then the decomposition of any gap into traffic, conversion, and average order value, because executives triage by factor (“we missed on conversion, not on basket”); then exceptions — the stores, categories, or campaigns whose deviation from expectation exceeds a threshold — rather than exhaustive listings. What is deliberately absent is as instructive: no models, no forecasts beyond plan, and very few adjectives.

Behind the page sits a verification routine that looks a great deal like Chapter 4's log. Before the report is released, the week's revenue is reconciled to the finance or order-management system of record; the row counts feeding each summary are compared against the prior week's, because a step change in volume is usually a pipeline failure rather than a demand event; the denominators are re-derived rather than inherited, so that a store closed mid-week does not quietly change what “comparable stores” means; and any definition that moved is annotated on the page itself. Trading analysts describe the job as protecting that page's precision — every rate's denominator stable week to week, every comparison like-for-like (this year's week aligned to the retail calendar, not the civil one), every “up 4 percent” decomposable on demand. The craft looks humble and is not: a trading report whose definitions wobble teaches an executive team to stop trusting numbers on Mondays, and the damage generalizes.

5.13.2 Healthy Averages Over a Deteriorating Base

The second vignette is a pattern subscription and repeat-purchase businesses rediscover regularly: healthy averages over a deteriorating base. A subscription-commerce company — the composite is generic, but the pattern is among the most cited reasons cohort tables exist — tracked overall monthly retention and average revenue per subscriber, and both lines held steady for quarters while the business quietly weakened. The mechanism was composition, exactly Section 5.5's: retention rises with subscriber tenure, so as acquisition slowed, the subscriber base aged, and the growing share of long-tenured, high-retention subscribers propped the overall averages while every new cohort retained worse than the one before it. The pooled metrics were true and reassuring; the cohort retention table — the same rows-by-age artifact as Table 5.7 — showed the down-and-to-the-right deterioration a year before the pooled lines bent.

What such a finding costs the analyst is worth describing, because it is rarely a single chart. Cohort deterioration reaches management as an escalation with a specific shape: the same-age comparison first (this quarter's cohort against the prior four at Month 3), then the reconciliation showing that the pooled metric is unchanged and why the two are not in conflict, then a decomposition of the deterioration into the channels or offers the weak cohorts came from, and only then a recommendation. The order matters because the pooled number is already in the room's memory, and an analyst who leads with “retention is falling” contradicts a dashboard the room trusts. The practice lesson lands on two of this chapter's disciplines at once: averages computed over shifting mixes answer questions nobody asked, and the same-age comparison is not an academic refinement — it is frequently the earliest honest signal a growth business gets.

5.13.3 Who Controls the Funnel's Definitions

The third vignette concerns the funnel — specifically, who controls its definitions. At streaming, e-commerce, and consumer-subscription firms, the conversion funnel is an operational institution: teams are named for stages, dashboards are organized by stage rates, and targets attach to stage boundaries. Which makes the boundaries themselves contested territory. What counts as a “visit” (does a bounce within five seconds count?), an “add to cart,” a “trial start” (card required or not?), an “active” account — each is an operational definition in Section 3.2's sense, each moves every rate downstream of it, and each has an owner with a target riding on it.

Mature organizations respond institutionally, and the mechanics are worth knowing before a new analyst is asked to change a definition. Stage definitions live in a governed dictionary with named owners; a proposed change is written as a diff against the current rule, accompanied by a backfill showing the affected rates under both definitions over a trailing window, so the size of the discontinuity is known before it appears; the change is approved by the metric's owner and the teams whose targets move; and on the day it ships, an annotation is written onto every chart that uses the rate, the way a store opening or a site migration would be. Teams that skip the backfill discover the change the way everyone else does — as a performance story that turns out to be an instrumentation story. The lesson reinforces rather than extends the chapter: funnels are powerful precisely because they are descriptive structure imposed on a journey, and the structure is only as trustworthy as the written definitions of its stages — whoever controls those definitions controls the funnel's story, which is Chapter 3's closing thread, operationalized.

5.13.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into an expectation worth internalizing before the job begins. Much of the analytical work organizations actually consume — the trading page, the cohort table, the funnel review, the profile behind a targeting brief — is descriptive, and the analysts who become indispensable are rarely the ones who reached for models fastest. They are the ones whose descriptions could be trusted blind: denominators named, centers matched to shapes, comparisons composition-checked, summaries that reconcile to certified totals, and narration that stops exactly where the evidence stops. Most delivered analytics value is descriptive, done precisely and narrated well — and precision plus narration is a career skill, not a stepping stone.

5.14 Ethics, Averages That Hide People

The Business Analytics in Practice section described description as the analytics organizations trust. This section examines the ethical weight of that trust, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), and cleaning as editorial power (Section 4.14) — to the summarizing step itself. The chapter's angle is specific: an average is a compression of people into a number, the compression always loses someone, and the ethical discipline is reporting heterogeneity honestly.

Begin with why this is an ethical matter and not merely a technical one. Every summary statistic in this chapter is a decision about whose experience the reported number represents. A mean order value is pulled upward by high-value baskets and, because it is computed at order grain, gives frequent purchasers more representation than infrequent ones; a median speaks for the middle and silences both tails; a pooled rate lets large groups speak for small ones; a “typical customer” profile manufactures a person who may not exist — the average of a bimodal

distribution describes nobody in either hump, which for StyleCraft means a composite of the urban loyalist and the suburban occasion shopper that matches neither. These compressions are unavoidable; description is compression. What is avoidable is presenting the compression as the population — the instinct to mistake a group's summary for its members that Rosling and colleagues catalog as the gap instinct, the human readiness to see two summary numbers as two homogeneous, separated groups when the underlying distributions overlap almost entirely (Rosling et al., 2018). Marketing runs on group summaries — segments, personas, metro comparisons — so marketing analysts hold this particular editorial power in unusual quantity, a concentration of influence continuous with the aggregation warnings of Section 3.3.1.

The risk concentrates where summaries steer treatment of people unevenly. Three of this chapter's own artifacts illustrate. The profile: a suburban-customer profile built on means invites a strategy addressed to the mean — and if suburban order values are wide or two-humped, a meaningful minority of suburban customers (the modest-basket regulars the format also serves) becomes invisible in every deck that quotes it, and decisions about staffing, assortment, and marketing are made as if they were not there. The pooled rate: Section 5.5's paradox is an ethics case as soon as the rate allocates anything — a channel or store group condemned on an aggregate rate that reverses within every stratum is a misallocation manufactured by arithmetic, and the burden lands on whoever staffed, stocked, or was employed by the condemned group. The concentration table: “the top decile carries the revenue” slides quickly into “the bottom deciles do not matter,” a sentence that pre-decides service levels for the majority of customers on a window-dependent ranking (Schmittlein et al., 1993) — defensible as strategy only if stated and examined, not smuggled as description.

The discipline this guide asks for is procedural, and it has three parts. First, spread travels with center: no group mean is reported for decision use without its dispersion and group size, and no rate is reported without its numerator, denominator, and underlying group count. Distributions that Section 4.9.1's shape checks show to be bimodal are shown, not averaged — a two-humped histogram in an appendix is more honest than any single row of summary statistics. Second, composition is checked and disclosed: every cross-group comparison of rates carries the one-line mix check of Section 5.5, and where stratification reverses or materially moves a headline, the stratified version is the headline. Third, the silenced are named: every summary that will steer differential treatment states, in a sentence, who it underweights — “this profile is order-weighted and speaks least for infrequent customers” costs a line and converts a hidden editorial choice into an inspectable one, exactly the conversion Section 4.14 demanded for exclusions. None of this pads the deliverable. All of it keeps faith with the people a certified table compresses — and with the decision-makers entitled to know whom their numbers are quietly not about.

CONCEPT

Report the People, Not Just the Average

Every group summary underrepresents someone. Before a descriptive artifact travels: pair every center with its spread and group size; show, rather than average over, distributions the shape checks flag as mixtures; run and disclose the composition check on every cross-group rate comparison; and state in one sentence whom the summary speaks for least. An average reported with these disclosures is description; reported without them, it is a decision about people who were never mentioned.

5.15 Chapter Summary

This chapter took the expansion review from one seductive number to a defensible account of what that number is made of, and built the descriptive disciplines the trip requires. The main point is that summarizing is the second analysis: every descriptive choice — denominator, center, grouping, percentage direction, comparison — changes the story a certified table tells without changing the table, so the choices are made explicitly, checked against composition, and reconciled to known totals rather than left to defaults.

The chapter's instruments arrived in sequence. Frequency tables and proportions established precise counting and the denominator problem — a share is a numerator-denominator pair, and the pair is named every time. The

choice of center was matched to distribution shape, mean and median reported together on skewed data, and the averaged average replaced by the weighted recombination that reconciles to totals. Spread was elevated to a finding: heterogeneity, especially the two-humped kind Chapter 4 flagged in StyleCraft's order values, is the question “is this one group?” in statistical clothes. Cross-tabulations brought the three percentage bases and the direction rule, and Simpson's paradox — computed, per this chapter's ownership, as the table-level case of Section 3.3.1's aggregation warning — made composition checks a standing obligation for every comparison of rates. Profiles assembled group comparisons on distributions at declared grains. The applied metric families organized the marketing repertoire — revenue, acquisition, conversion, retention, engagement, formulas cataloged in Appendix B — and the decomposition identity did the chapter's central work: average order value factored exactly into basket size and revenue per unit, revenue per unit examined against category mix, and the whole audited by the twin checks that factors multiply back and summaries recombine to certified totals. Period-active cohort retention tables put comparisons at like ages, while naming what alignment does not remove; the funnel disciplined description along the journey and exposed stage definitions and stage units as measurement decisions; concentration analysis, benchmarks, and the insight statement turned findings into context-bearing sentences that stop where description stops. Throughout, AI assistants drafted the summaries while the analyst audited the claims — denominators, centers, compositions, reconciliations, and verbs — and the ethics section named the stakes: averages hide people, and honest heterogeneity reporting is what keeps the compression accountable.

Looking ahead, the labs left StyleCraft with a decomposed AOV gap, a profile of two customer worlds, and cohort evidence about how each world repeats — description that keeps insisting the company's customers are not one population. The next chapter takes that insistence formally: it opens Part II by asking which customer groups the data actually contains, building the customer-grain feature table and using recency, frequency, and monetary structure — and then clustering — to discover segments rather than assume them. Description found the humps; segmentation names them.

5.16 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: name the denominator, match the center to the shape, check composition before comparing, decompose before interpreting, and reconcile every summary to known totals. They are organized into two groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Self-Assessment #3 draws on the required-practice sets. In-class activities are prepared for discussion rather than submitted. Each exercise also carries its assignment label — required practice, homework submission, or in-class discussion — so that instructors can assign selectively.

5.16.1 Core Chapter Practice

Exercise 5.1 Concept Check (Required Practice)

12. State the denominator problem in one sentence, and give two denominators that turn the same email-open numerator into materially different rates.
13. On a right-skewed order-value distribution, state the order in which mean, median, and mode commonly fall, explain why that ordering is a working heuristic rather than a theorem, and give one business question each center is the right answer to.
14. Why is the simple average of per-store AOVs generally not the company's AOV, and what recombination is?
15. Two stores share a mean order value of $60. Describe two spread patterns that would demand different marketing responses, and name the statistics that distinguish them.
16. A cross-tabulation supports three percentage bases. Name them, compute all three for the 540-customer cell of Table 5.3, and state Zeisel's direction rule for choosing which to report.
17. Define Simpson's paradox without the word “paradox,” and state the one-line check that detects the conditions for it.

18. Write the decomposition identity for AOV, extend it by one factor to revenue per customer, and state the two verification equalities every decomposition must pass. Then explain why “basket size accounts for 60 percent of the gap” is not a claim the identity supports.
19. Distinguish a period-active cohort table from a survival retention table, state which one can show a rise from one period to the next and why, and explain what reading down a column compares that the pooled rate cannot — and what it still does not control for.

Exercise 5.2 Pick the Denominator (Required Practice)

For each reported figure, name at least two defensible denominators, state how the choice changes the claim, and pick one with a one-sentence justification tied to a decision.

18. “Our SMS click rate is 9 percent.”
19. “Store accounts for 44 percent of our business.”
20. “Repeat rate at the Greenwich store is 22 percent.”
21. “The Welcome Series converts at 3.1 percent.”
22. “Occasionwear is 60 percent of suburban sales.”
23. “CAC came in at $41 last quarter.”

Exercise 5.3 Choose the Center and Report the Spread (Required Practice)

For each situation, choose the center (or pair) to report, the spread measure to accompany it, and state in one sentence what the choice protects the reader from.

12. Order value across all channels, for the trading report's headline.
13. The “typical basket” for a store-associate training deck.
14. Units per order, which takes only small integer values.
15. Revenue per customer for the expansion review, where Chapter 4's exploration showed a two-humped shape.
16. Time between first and second purchase, where a third of customers have no second purchase yet.

Exercise 5.4 Reconcile the Summary (Required Practice)

A certified file has 80,000 order lines, 52,000 orders, 8,000 customers, and total revenue of $4,160,000. An AI-drafted summary reports: metro AOVs of $71 (urban, 40,000 orders) and $118 (suburban, 11,600 orders); an overall AOV of $94.50 described as “the average of the metro AOVs”; and a repeat rate of 41 percent “of active customers.” Write the reconciliation audit in advance: which totals the metro rows must sum to and whether they do; the correct pooled AOV from the certified totals and the size of the averaged-average error; what is wrong, per Section 3.5.1, with the repeat-rate sentence as reported; and the two questions you would ask before this summary is allowed into a pre-read. (This exercise applies the chapter's verification theme; it is the summaries' version of Exercise 4.4.)

Exercise 5.5 Decompose the Gap End-to-End (Homework Submission)

Complete Lab 5.1 Part B on the certified full file: the metro AOV table with reconciliation to certified totals, the two-step analysis (the exact basket × revenue-per-unit decomposition, then the standardized comparison of revenue per unit at common within-category prices, plus the same-product realized-price check the miniature could not perform), the identity audit, and a five-sentence insight statement per Section 5.9 addressed to the expansion review. Submit the notebook, the reconciliation log, and the completed AI-Use Appendix (Appendix D) recording every assistant exchange, including at least one delegated summary whose stated denominator, center, or reconciliation you had to correct.

Exercise 5.6 AI-Assisted Summary Audit (Homework Submission)

Give an AI assistant the Lab 5.1 Part A miniature and this deliberately vague prompt, verbatim: “Summarize this sales data and tell me what it means.” Run its code on a copy. Then audit against the chapter: identify every center it chose and whether the shape justified it; every rate or share it reported and whether the denominator was stated;

whether any overall figure was an averaged average; whether its totals reconcile to the miniature's certified figures (14 lines, 8 orders, 660.0); and every causal verb in its narration, each rewritten into descriptive language. Conclude with a rewritten prompt per the AI in Practice pattern that would have forced the choices into the open, and submit the audit with the AI-Use Appendix.

5.16.2 In-Class Activities

Exercise 5.7 Find the Flaw in the Description (In-Class Discussion)

Each scenario below describes a summary that is computed correctly. Find the flaw, name the section whose discipline it violates, and propose the repair.

8. A deck reports “our customers' average age band is 25–34” from the modal age_band, and a second slide computes a numeric “mean age” from the band midpoints.
9. The overall conversion rate in a pre-read is the simple average of forty campaigns' conversion rates, and it exceeds every large campaign's rate.
10. A channel comparison shows App with the highest repeat rate; App launched fourteen months ago and its customers skew to the newest signups' most engaged adopters.
11. A funnel review celebrates checkout conversion rising from 58 to 66 percent in the week a new definition stopped counting abandoned guest sessions as checkouts.
12. A campaign dashboard reports a single “funnel conversion rate” computed as attributed orders divided by impressions, and labels the three stages a nested funnel.
13. A profile of “the suburban customer” reports means only; the order-value histogram for the metro shows two humps.
14. The top-decile revenue share is reported from a 30-day window to argue that “a tiny elite carries the company,” and the deciles were cut on customers active in that window only.

Exercise 5.8 Averages That Hide People Mini-Cases (In-Class Discussion)

For each mini-case, identify who the summary hides, what Section 5.14's discipline requires before the artifact travels, and what one disclosure sentence you would attach.

8. A staffing model allocates store hours by mean transaction time per store; two stores serve a bimodal mix of quick pickup orders and long occasionwear fittings.
9. A “customers like these don't respond” conclusion about a metro, drawn from a pooled response rate, when response within every age band in that metro matches the company average.
10. A persona deck describes “the StyleCraft customer” as a 26-year-old app user, built from modal values of every attribute independently.
11. A proposal to cut service investment for bottom-decile customers, ranked on a quarter of revenue, presented without the ranking's window or churn-of-decile-membership.
12. An executive asks for “just the one number” for suburban performance, and the analyst must choose what to send and what to attach.

5.17 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Benchmark. A reference value — internal-historical, internal-cross-sectional, or external — against which a metric is placed so that the number becomes a claim; external benchmarks rarely share a metric's four defining elements and are used as directional context only.

Central tendency. The family of single-value summaries of a numeric variable — mean, median, and mode — whose members diverge on skewed data, making the choice among them an analytic decision (adapted from Freedman et al., 2007).

Cohort. A group of units defined by a shared starting event in a shared time window, followed forward from that event so that groups are compared at the same age rather than the same date; alignment removes tenure from the comparison but not calendar-period or compositional differences (adapted from Glenn, 2005).

Cohort retention table. The descriptive artifact of cohort analysis: rows for cohorts, columns for periods since the starting event, cells holding each cohort's share still present under a stated activity rule; read down a column for same-age comparisons. A period-active table records activity in each individual period and may rise from one period to the next; a survival retention table records continuous retention and is non-increasing by construction.

Concentration analysis. The description of how unevenly a total is distributed across the units producing it, typically via ranked deciles and their shares; its summary claims are window-dependent (adapted from Twedt, 1964; Schmittlein et al., 1993).

Conversion funnel. A descriptive structure arranging a journey as ordered stages with counts at each stage and rates between stages, each stage bounded by a stated operational definition and each rate labeled step-to-step or cumulative; stages are strictly nested only when every stage counts the same unit (adapted from Strong, 1925; Lavidge & Steiner, 1961).

Cross-tabulation. A table counting observations for every combination of two categorical variables, supporting cell, row, and column percentages; percentages are computed in the direction of the comparison being made (adapted from Zeisel, 1985; Agresti, 2019).

Customer profile. A curated table of descriptive comparisons across groups — sizes first, centers with spreads, each cell at a declared grain — chosen to answer an analytic specification rather than to exhaust the data.

Decomposition identity. An exact factoring of a composite metric into components that multiply back to it — for example, average order value into units per order times revenue per unit — used to express a gap as a product of factors and audited by recombination against known totals. It yields multiplicative factors, not additive percentage shares.

Denominator problem. The failure mode in which a proportion, rate, or average is reported or read without its denominator, allowing several correct numbers to support several different claims about the same numerator.

Dispersion. The family of summaries of a variable's variability — range, interquartile range, and standard deviation — one of which accompanies every reported center (adapted from Freedman et al., 2007; Tukey, 1977).

Frequency table. A summary listing each distinct value of a variable with its count and, in working form, its proportion of a stated total, complete only with its unit of analysis and denominator declared (adapted from Freedman et al., 2007; Agresti, 2019).

Heterogeneity. Wide or structured variation within a group — including bimodality — treated in this guide as a descriptive finding prompting the question of whether the group is one population, rather than as noise around an average.

Insight statement. A descriptive claim equipped for decision use: the comparison or decomposition that gives the number meaning, magnitude in decision units, connection to the named decision, and verbs that assert no more than the description supports.

Simpson's paradox. The reversal or disappearance of an association between two variables upon disaggregation by a third, arising when compared groups are unevenly composed across strata with different base rates; detected by composition checks, not arithmetic rechecks (adapted from Simpson, 1951; Blyth, 1972).

Weighted mean. The recombination of group means into an overall mean using group sizes as weights; the only recombination that reconciles to the pooled total, and the standing correction to the averaged average.

5.18 Further Readings

Students who want additional background may begin with the following readings. General descriptive foundations are listed first, marketing applications second.

- Freedman et al. (2007) for the clearest available treatment of summary statistics, what they assume, and how they mislead — the statistical conscience behind Sections 5.2 through 5.4.
- Zeisel (1985) for the classic, still-unmatched short book on percentages, tables, and the direction rule, and Blyth (1972) for a compact formal treatment of Simpson's paradox; readers wanting the original note may consult Simpson (1951).
- Farris et al. (2010) for the standard practitioner reference on marketing metrics — the extended companion to Section 5.7 and Appendix B.
- Schmittlein et al. (1993) for the definitive skeptical examination of 80/20 claims, including why measured concentration depends on the observation window.
- Glenn (2005) for a rigorous short treatment of cohort analysis and the identification problems that make same-age comparison necessary — and that same-age comparison does not fully solve.
- Rosling et al. (2018) for the gap instinct and the broader case that summary comparisons of groups systematically overstate their separation — the accessible companion to Section 5.14.

5.19 References

Agresti, A. (2019). *An introduction to categorical data analysis* (3rd ed.). Wiley.

Blyth, C. R. (1972). On Simpson's paradox and the sure-thing principle. *Journal of the American Statistical Association, 67*(338), 364–366. https://doi.org/10.1080/01621459.1972.10482387

Farris, P. W., Bendle, N. T., Pfeifer, P. E., & Reibstein, D. J. (2010). *Marketing metrics: The definitive guide to measuring marketing performance* (2nd ed.). Pearson Education.

Freedman, D., Pisani, R., & Purves, R. (2007). *Statistics* (4th ed.). W. W. Norton.

Glenn, N. D. (2005). *Cohort analysis* (2nd ed.). Sage.

Lavidge, R. J., & Steiner, G. A. (1961). A model for predictive measurements of advertising effectiveness. *Journal of Marketing, 25*(6), 59–62. https://doi.org/10.1177/002224296102500611

Rosling, H., Rosling Rönnlund, A., & Rosling, O. (2018). *Factfulness: Ten reasons we're wrong about the world — and why things are better than you think*. Flatiron Books.

Schmittlein, D. C., Cooper, L. G., & Morrison, D. G. (1993). Truth in concentration in the land of (80/20) laws. *Marketing Science, 12*(2), 167–183. https://doi.org/10.1287/mksc.12.2.167

Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society: Series B (Methodological), 13*(2), 238–241. https://doi.org/10.1111/j.2517-6161.1951.tb00088.x

Strong, E. K. (1925). *The psychology of selling and advertising*. McGraw-Hill.

Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley.

Twedt, D. W. (1964). How important to marketing strategy is the “heavy user”? *Journal of Marketing, 28*(1), 71–72. https://doi.org/10.1177/002224296402800117

Zeisel, H. (1985). *Say it with figures* (6th ed.). Harper & Row.

Chapter 06

Segmentation and Targeting Analytics

Discovering, Verifying, and Targeting Customers

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter opens Part II with customer segmentation and the guide's first sustained application of unsupervised learning: the move from describing a customer base to dividing it into groups a marketing organization can act on. It places segmentation inside the segmentation-targeting-positioning framework, surveys segmentation bases and the data each requires, and separates a priori schemes declared before analysis from post hoc schemes estimated from data. Its two working methods are RFM scoring on an exhaustive named grid and k-means clustering, developed from distance and standardization through the choice of k and the profiling, stability, and holdout checks that separate a defensible segmentation from a plausible-looking accident. Useful-segment criteria and a worked attractiveness matrix turn verified segments into targeting decisions. The labs build StyleCraft's customer-feature table, score the grid, cluster at full scale, and test the recovery of four designed segments.

KEYWORDS

market segmentation; targeting; stp framework; rfm analysis; customer features; k-means clustering; standardization; silhouette score; cluster validation; unsupervised learning

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Segmentation and targeting analytics. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 6, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas and NumPy are sponsored projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. scikit-learn is a community project supported by the scikit-learn consortium. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the

underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

- Explain the strategic role of segmentation in marketing decision-making, situate it within the segmentation-targeting-positioning framework, and distinguish segmentation as a strategic act from segmentation as a computation.
- Compare the major segmentation bases — geographic, demographic, psychographic, and behavioral — and state the data each requires, including which bases StyleCraft's tables can and cannot support.
- Distinguish a priori from post hoc segmentation, and state precisely what a post hoc method estimates from data and what the analyst must still supply.
- Conduct and interpret an RFM analysis: define recency, frequency, and monetary value at the customer grain, score them on a declared rule, resolve tied values defensibly, and read an exhaustive named grid.
- Build a customer-grain feature table from transaction data, declaring the population, the analysis date, the window, and the derivation rule for every feature, and reconcile the table to certified totals.
- Explain why distance-based methods require standardization, demonstrate with hand arithmetic how unscaled units let one feature dominate a distance, and explain why standardization does not address redundancy among features.
- Apply k-means clustering for exploratory segmentation: sketch the algorithm, explain initialization sensitivity, and choose k using the elbow method, silhouette scores, and business judgment together.
- Profile clusters at declared grains, measure stability with a permutation-invariant index across seeds and subsamples, profile withheld attributes, and state precisely what each check does and does not establish.
- Evaluate whether segments are useful against the five criteria — measurable, substantial, accessible, differentiable, actionable — and build a segment-attractiveness matrix that marks evidence separately from judgment.
- Identify where segmentation crosses ethical lines: sensitive attributes and their proxies, differential pricing and offers, and treatment decisions that punish tenure artifacts rather than customer potential.

Chapter 5 closed Part I with a description that kept insisting on one thing: StyleCraft's customers are not one population. The decomposed average-order-value gap, the two-grain profile, and the cohort tables all pointed to distinct customer worlds living inside a single company — description found the humps. This chapter, which opens Part II, names them. It develops segmentation as the discipline of dividing a customer base into groups that can be measured, verified, and — the part that makes it strategy rather than arithmetic — treated differently. The route runs from strategic foundations (why segment, on what bases, decided in advance or estimated from data) through the two working methods of the analyst (RFM scoring and k-means clustering), and ends where segmentation earns its keep: deciding which segments to target and with what. Because the groups in this chapter are estimated by an algorithm rather than declared by managers, the chapter also installs a new verification obligation — the audit of a clustering solution — that the AI section, the labs, and the exercises all rehearse.

CONCEPT

What This Chapter Is Really About

Chapter 5 argued that summarizing is the second analysis. This chapter makes the argument that follows from it: grouping is the third, and it is the first analysis in this guide whose output is produced by an algorithm rather than by arithmetic the analyst fully controls. A frequency table cannot be wrong about which rows exist; a clustering can — it will return as many groups as it was asked for, drawn with perfect confidence, whether or not those groups exist in any sense a marketer should care about. The discipline of Part I was reconciliation: check every summary against known totals. The discipline this chapter adds is validation: an algorithmic grouping is a hypothesis about structure, and it earns trust only by surviving tests it was not built to pass — refits from new starting points and new samples, profiling on variables it never saw, and the blunt managerial question of whether the groups can be treated differently at all. A segmentation that survives those tests changes the marketing calendar. One that is merely admired in a deck changes nothing, however elegant its geometry.

6.1 Marketing Decision Context: Four Slots on the Fall Calendar

Chapter 5 ended with the expansion review in possession of a defensible account of its one seductive number: the suburban average order value decomposed into basket size and occasionwear mix, a two-grain profile showing two customer worlds, and cohort tables comparing those worlds at the same age. The review's capital-allocation question moves to later chapters. But the meeting produced a second decision, quieter and nearer, and it lands on the CRM manager — the same CRM manager whose worry about declining repeat purchase opened this guide in Chapter 1.

The decision is the fall lifecycle-marketing program. StyleCraft's email and app platform can operationally sustain four new differentiated customer journeys this fall — four sequences of messages, offers, and triggers, each with its own creative, its own cadence, and its own budget line. Not seven, not nine: four is what the CRM team can build, test, and maintain alongside the weekly drops. Two clarifications of that constraint matter later and are stated now. The always-on welcome program that greets a new signup and works toward a first purchase is existing business as usual; it continues, and it does not consume one of the four new slots. And a slot buys one journey, not one message: a journey may carry an internal branch keyed to a declared rule, at the cost of the extra creative that branch requires, but a fifth creative-and-measurement frame is what the platform cannot carry. Every one of StyleCraft's purchasing customers must be assigned to exactly one of the four journeys by a rule that lives in the database rather than in someone's judgment, every signed-up customer who has not purchased must be accounted for somewhere, and the program locks in twenty-one days, when creative production begins. The stake is the retention budget: after Chapter 5's cohort evidence, no one in the building still believes one message serves every customer, and the money will follow whatever map of the customer base the CRM manager brings to the sign-off meeting.

The problem is that three maps have already arrived, and they disagree. The brand agency's pitch deck contains seven personas — vivid, named, photographed characters (“The Weekend Curator,” “The Campus Tastemaker”) built from two focus groups and considerable craft, none of them connected to a single row of StyleCraft's data. The head of retail wants a simpler map: two segments, urban and suburban, straight from Chapter 5's profile, with a “suburban VIP” push to court the high-basket occasion shopper. And a data-science-inclined intern, working overnight with an AI assistant, has produced the third map: a nine-cluster solution on customer data, each cluster already named by the assistant with unsettling fluency, presented in a notebook nobody has audited. Nine clusters, seven personas, two metros — and four slots on the calendar.

The CRM manager's task is therefore not to pick the most persuasive deck. It is to answer, with evidence, the question all three maps claim to answer: which customer groups does StyleCraft's data actually contain? That question decomposes into the questions this chapter teaches. First, the strategy question: what should the groups be based on — who customers are, where they live, or what they do — given that the fall program can only act on

differences it can reach? Second, the discovery question: should the groups be declared in advance, the way the head of retail's two-metro scheme declares them, or estimated from the data, the way the intern's clustering claims to have discovered them — and if estimated, by what method and with what checks? Third, the verification question, which the intern's notebook makes urgent: an algorithm asked for nine clusters produced nine clusters, as it would have produced six or thirteen — what would it take to establish that any clustering describes real structure rather than an arbitrary partition wearing confident names? And fourth, the targeting question: even among real segments, four slots force choices — which groups justify a differentiated journey, and by what criteria?

The deadline is fixed, the audience includes the head of retail and the agency, and the deliverable is concrete: at most four segments, each defined by a written rule on StyleCraft's own columns, each profiled honestly per Chapter 5's disciplines, each verified to be more than an artifact, and each attached to a recommended treatment. The chapter builds toward that deliverable in order. Sections 6.2 through 6.4 establish the strategic frame: what segmentation is for, what it can be based on, and the a priori versus post hoc distinction that separates the head of retail's map from the intern's. Sections 6.5 and 6.6 build the analyst's first method and its raw material: RFM analysis and the customer-grain feature table it requires. Sections 6.7 through 6.10 develop clustering — distance, scaling, k-means, the choice of k, and the profiling and stability checks that make a clustering defensible. Section 6.11 converts segments into targeting decisions. Section 6.12 examines how AI assistants help and fail at exactly this work, and the labs in Section 6.13 run the full sequence on StyleCraft's certified data — where, by design, the right answer is known: the dataset was built to contain four segments, and the analysis must find them.

6.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The agency's seven personas, the head of retail's two metros, and the intern's nine clusters are all segmentations of the same customer base. For each map, state what it is based on and what data, if any, could prove it wrong.
- The platform supports four journeys. Suppose the data genuinely contained six well-separated customer groups. What are the CRM manager's options, and what does each cost?
- The intern's assistant named a cluster “Affluent Suburban Loyalists.” StyleCraft's customers table contains no income column, and Chapter 5 showed suburban shoppers rarely enroll in loyalty. What has the assistant done, and which section of this chapter is designed to catch it?
- The head of retail's two-metro scheme requires no algorithm at all: every customer's metro is already a column. What does that scheme gain by being so simple, and what did Chapter 5's profile suggest it loses?

6.2 Segmentation as Strategy and Segmentation as Computation

Everything in this chapter serves one strategic idea, and it is older than any algorithm in it. A market is not a homogeneous mass of interchangeable buyers; it is a collection of groups whose wants, behaviors, and responses to marketing differ, and a firm that recognizes those groups can serve chosen ones deliberately rather than serving an imaginary average customer badly (Smith, 1956). Chapter 5 supplied this guide's empirical version of the same idea: heterogeneity is a finding, and StyleCraft's two-humped order values were the question “is this one group?” in statistical clothes. Segmentation is the affirmative program built on the answer no.

DEFINITION

Market Segmentation

Market segmentation is the division of a market into distinct groups of buyers — segments — that are internally similar and externally different on characteristics relevant to how the firm will serve them, undertaken so that marketing effort can be differentiated across groups rather than uniform over the whole market.

Source: Adapted from Smith (1956) and Kotler and Keller (2016).

Segmentation is the first move of a three-part framework that organizes marketing strategy: segmentation, targeting, and positioning. The firm first divides the market into segments; it then targets — selects which segments to serve, with what priority and resources; and it finally positions — designs an offering and message intended to occupy a distinct place in the targeted segment's mind (Kotler & Keller, 2016). This guide is an analytics guide, so it engages the framework from the analyst's seat: segmentation is where the analytics is densest and is this chapter's technical core; targeting is a decision this chapter equips with evidence in Section 6.11; and positioning is a creative and strategic act the analyst informs — with segment profiles — but does not compute. When the CRM manager assigns thousands of customers to four journeys, all three moves happen: the base is segmented, four groups are targeted with differentiated treatment, and each journey's creative positions StyleCraft for its group.

DEFINITION

Segmentation, Targeting, and Positioning (STP)

STP is the sequential strategy framework in which a firm divides a market into segments, selects target segments to serve, and positions its offering distinctively for each target. In analytics practice, segmentation is computed, targeting is decided on computed evidence, and positioning is informed by segment profiles.

Source: Adapted from Kotler and Keller (2016).

The working distinction the rest of the chapter depends on is between segmentation as a strategic act and segmentation as a computation, because the two are routinely confused and the confusion is expensive in both directions. As a strategic act, segmentation is a commitment: it declares which customer differences the organization will organize itself around — whose journeys get built, whose offers differ, whose names appear on the marketing calendar. As a computation, segmentation is a procedure: a scoring rule or a clustering algorithm partitions rows of a customer table. The computation is the easy part, and it has never been easier — Section 6.12 will show that a fluent cluster solution is now one prompt away. The strategy is the binding part: a partition of rows becomes a segmentation only when each group is attached to a difference in treatment the organization can actually execute. The intern's nine clusters may be excellent geometry; against a four-journey calendar, at least five of them are, strategically, not segments at all. The reverse failure is just as real: a segmentation declared purely from the executive floor — two metros, because the org chart has an urban lead and a suburban lead — commits the organization around a difference the data may show is not the difference that matters.

CONCEPT

Segmentation Is a Treatment Decision

A partition of customers is arithmetic. A segmentation is a partition plus a commitment: for each group, a way the organization will treat it differently — different journeys, offers, budgets, service levels, or products. Before admiring any segmentation, ask the treatment question: what will we do differently to each of these groups that we could not do to the whole base? Groups without an answer are analysis; groups with an answer are strategy. The question also runs backward, and Section 6.11 will formalize it: the number of treatments the organization can execute is a design constraint on the segmentation itself — a constraint-as-design-input in exactly the sense of Section 2.4.

Source: Course concept developed for this guide, informed by Kotler and Keller (2016) and Yankelovich and Meer (2006).

The distinction settles the opening case's first argument before any data is touched. The agency's personas are positioning aids — vivid characters that help creative teams write to someone rather than to no one — and nothing in this chapter retires them from that job. But they are not, as delivered, a segmentation: no rule assigns a StyleCraft customer to “The Weekend Curator,” so no journey, budget, or measurement can attach to it. The head of retail's

two-metro scheme is a genuine segmentation — every customer is assigned by rule, and treatments can differ by metro — but whether it is a good one depends on whether metro is the difference that best predicts differential response, which is an empirical question the scheme itself cannot settle. And the intern's clusters are, so far, a computation awaiting the strategic test. The chapter's job is to give the CRM manager the machinery to run all three maps through the same discipline.

6.3 Segmentation Bases and the Data They Require

If segmentation divides customers by their differences, the first design decision is which differences. The candidate characteristics are called segmentation bases, and marketing practice organizes them into four families (Kotler & Keller, 2016; Wedel & Kamakura, 2000). Geographic bases divide by location: region, metro, urban versus suburban, climate. Demographic bases divide by who customers are in census terms: age, generation, gender, income, household composition. Psychographic bases divide by how customers think and live: values, lifestyles, attitudes, interests. Behavioral bases divide by what customers do with the category and the brand: purchase recency and frequency, spend, product mix, channel use, loyalty status, response to promotions.

DEFINITION

Segmentation Base

A segmentation base is the characteristic or set of characteristics on which customers are divided into segments — geographic, demographic, psychographic, or behavioral. The choice of base determines what data the segmentation requires, what differences it can express, and what treatments it can support.

Source: Adapted from Wedel and Kamakura (2000) and Kotler and Keller (2016).

For the analyst, the base decision is inseparable from a data decision, because each family makes different demands on the tables of Chapter 3. Table 6.1 walks the four families through StyleCraft's actual columns, and the exercise generalizes: before endorsing any base, locate the columns that would operationalize it, at the grain the segmentation needs, with the validity Chapter 3 taught you to interrogate.

Table 6.1

Segmentation bases and their data requirements at StyleCraft

Base family	Example variables	StyleCraft columns	Data notes for the analyst
Geographic	Region, metro, store proximity	home_region, home_metro, primary_store_id (customers)	Available for every customer; coarse; stable; a proxy for much it does not name (Section 6.16)
Demographic	Age, generation, gender	age_band, generation, gender (customers)	Self-reported at signup; age_band optional, so partly missing; no income column exists — “affluent” is inferred, not measured
Psychographic	Values, lifestyle, style identity	none	Not in the schema; requires surveys or licensed data; the agency’s personas live here, unmeasured
Behavioral	Recency, frequency, spend, discount use, channel, loyalty	derived from transactions; loyalty_tier, app_user, email_opt_in (customers)	Richest family; requires the grain change of Section 6.6 — behavior lives in transactions at line grain and must be rolled up to customers

Two lessons from the table organize what follows. The first is that behavioral bases dominate database marketing practice, and for a principled reason: they are computed from what customers verifiably did rather than from what customers say or from what their zip code implies, and differential response to marketing — the thing

targeting cares about — is itself a behavior. The empirical segmentation literature reaches a similar verdict: bases closest to purchase behavior tend to be the most actionable for customer-level marketing, while general characteristics like demographics travel better for media buying and product design than for predicting response (Wedel & Kamakura, 2000; Haley, 1968). The fall program is customer-level marketing on StyleCraft's own base, which is why this chapter's two working methods — RFM in Section 6.5 and clustering on behavioral features in Sections 6.7 through 6.10 — are both behavioral.

The second lesson is that the base decision is also an honesty decision. StyleCraft's schema can support geographic, demographic, and behavioral segmentation at declared levels of quality; it cannot support psychographic segmentation at all, and it cannot support "affluent" as anything but an inference from geography. A segmentation deliverable that names segments in psychographic or income language — as the intern's assistant did — is asserting bases the data does not contain, a validity failure in exactly Section 3.6's sense. The discipline is to name segments in the vocabulary of the base that produced them, a rule Section 6.10 will make operational.

Well-chosen bases are also frequently combined. Nothing restricts a segmentation to one family: the scheme the labs build clusters on behavioral features and then profiles the resulting groups geographically and demographically — using the other families as description and as a check rather than as inputs. That division of labor, behavioral bases in, other bases held out for profiling, is deliberate, and Section 6.10 will show both what it can establish and what it cannot.

6.4 Two Roads to Segments: A Priori and Post Hoc

With a base chosen, there remain two roads to actual segments, and the segmentation literature has named them for half a century (Wind, 1978). In a priori segmentation, the analyst declares the segments in advance: the number of groups and the rule that assigns customers to them are decided before the data are consulted, and the data's job is to size and profile groups whose definitions were never in question. Urban versus suburban metro is a priori; so is loyalty tier, so is "customers acquired through TikTok," and so is every RFM grid cell in Section 6.5, because the scoring rule and the named grid are fixed before any customer is scored. In post hoc segmentation, the segments are estimated from the data: the analyst chooses the features, the method, and the candidate numbers of groups, and the data determine which customers belong together and where the boundaries fall. Clustering is the canonical post hoc method, and Sections 6.7 through 6.10 are its treatment.

DEFINITION

A Priori and Post Hoc Segmentation

A segmentation is a priori when the number of segments and the assignment rule are declared before analysis, so that data serve to size and profile predefined groups. A segmentation is post hoc when group membership and boundaries are estimated from the data after the analyst specifies the features, the method, and the candidate values of k. The post hoc road removes the analyst's prior about who belongs with whom; it does not remove the analyst's choices about what to measure, how to scale it, and how many groups to ask for. In k-means, k is selected using evidence and judgment (Section 6.9); it is not produced by the algorithm.

Source: Adapted from Wind (1978); k-means characterization per Pedregosa et al. (2011).

Neither road is superior; they answer different questions and fail in different ways, and Table 6.2 sets the comparison the CRM manager needs. The a priori road is transparent, cheap, instantly explainable, and guaranteed actionable if the defining variable is one the organization can act on — a journey keyed to loyalty_tier can be built this afternoon. Its risk is exactly its virtue: the segments encode a prior belief, and if the belief is wrong — if the difference that matters is not the one declared — the data will obligingly size and profile groups that miss the structure. Chapter 5's profile hinted at precisely this: the urban-suburban split is real, but the spread within each metro was wide, which is the descriptive signature of structure the two-metro scheme cannot express. The post hoc road can find structure nobody declared — that is its entire value — but it trades away transparency: it requires the

feature table of Section 6.6, the scaling discipline of Section 6.7, an algorithm with sensitivities of its own, and, above all, the validation of Section 6.10, because a method that returns groups will return them whether or not they exist.

Table 6.2

A priori versus post hoc segmentation

Dimension	A priori	Post hoc
Segments defined	Before analysis, by declared rule	By the analysis, from the data, given the analyst's inputs
Number of segments	Chosen by the analyst or given by the variable	Supplied by the analyst as k and defended on evidence (Section 6.9)
Typical methods	Cross-tabs, profiles, RFM scoring	Cluster analysis (k-means in this guide)
Chief strength	Transparent, explainable, immediately actionable	Can find structure no one declared
Chief risk	Encodes a possibly wrong prior belief	Returns a partition whether or not structure exists; requires validation (Section 6.10)
Verification burden	Sizing and profiling done honestly (Chapter 5)	Everything in Chapter 5, plus stability and holdout checks
Opening-case example	Head of retail's two metros	Intern's nine clusters

The two roads also combine, and the combination is this chapter's recommended practice rather than a compromise. A common professional sequence runs post hoc discovery first, a priori operation second: clustering estimates candidate structure; the analyst translates the resulting groups into explicit written rules on named columns and retains the fitted model that produced them (a step Section 6.10 formalizes); and the organization then operates the segmentation a priori — every new customer assigned by a stated procedure, every journey keyed to that assignment, every assignment auditable. The discovery is statistical; the deployment is governance. This sequence is what the labs perform, and it is what the CRM manager's deliverable — "each segment defined by a written rule on StyleCraft's own columns" — actually requires.

One more distinction keeps the roads honest, and it echoes Chapter 5's insistence that a designed answer key is not a reason to skip the work. StyleCraft's dataset was built, per its specification, to contain four latent customer segments. That fact makes the labs gradable: the post hoc analysis has a known truth to recover. It does not make the analysis circular, because the analyst's procedure — features, scaling, method, k, validation — must earn the recovery; a bad procedure will happily fail to find structure that is genuinely there, and the exercises include exactly such failures to diagnose. Real customer bases do not ship with answer keys, which is why the validation discipline of Section 6.10, rehearsed here where the truth is known, is the portable skill.

6.5 RFM Analysis: The Workhorse of Behavioral Segmentation

The analyst's first working method predates every algorithm in this chapter and remains one of the most operationally used segmentation techniques in direct and database marketing: RFM analysis, the scoring of customers on recency, frequency, and monetary value (Hughes, 1994). The three quantities themselves are not new to this guide — recency, frequency, and monetary value are metrics, defined per the four-element discipline of Section 3.5.1 and cataloged in Appendix B; Chapter 5 used all three descriptively. What this chapter owns is the method built on them: scoring each quantity into ranked bands, combining the scores into a grid, and naming the grid's regions as operable segments.

The three definitions must be fixed before any scoring, because each hides the measurement decisions Chapter 3 taught you to write down. Recency is the elapsed time since the customer's most recent purchase, computed against a declared analysis date — this guide's labs use June 30, 2026, the same analysis date as Lab 3.1 — and stated in days. Frequency is the count of the customer's distinct orders in a declared window; this chapter uses the

full 24-month file and counts orders, not lines, which is a grain commitment (an order of five lines is one act of purchasing, not five). Monetary value is the customer's total revenue in the same window; a common variant uses average order value instead, and the choice matters — total revenue rewards frequent modest baskets, average order value rewards rare large ones, and on StyleCraft that is precisely the urban-suburban contrast — so the choice is declared, not defaulted. Every one of these is a numerator, denominator or window, and filter decision, and the standing rule of Section 3.5.1 applies: write the definition before computing the number.

DEFINITION

RFM Analysis

RFM analysis is a behavioral segmentation method that scores each customer on recency of last purchase, frequency of purchases, and monetary value in a declared window — classically by quintile — and groups customers by their combination of scores. The scoring tradition rests on the practitioner premise that recent, frequent, high-value buyers are the most responsive to direct marketing; treated analytically, each grid region's treatment logic is a hypothesis about response, to be tested by campaigns rather than assumed from the score.

Source: Adapted from Hughes (1994). For the formal link between RFM measures and customer value, see Fader et al. (2005).

The scoring step converts each raw quantity into a rank-based score, classically by quintiles: customers are sorted on the quantity and divided into five equal-count bins, scored 5 for the best bin through 1 for the worst — where “best” means most recent, most frequent, and highest value, so recency is scored in reverse of its raw days. Quintile scoring has two properties that explain its persistence. It is distribution-free: Chapter 5 showed that spend and frequency are strongly right-skewed and concentration-heavy, and ranking neutralizes the skew that would make raw-value cutoffs arbitrary. And it is self-calibrating within its population: the bins move with the base, so “a 5 on monetary” means the top fifth of this base in this window rather than a fixed dollar threshold that inflation and growth quietly retire.

That second property is regularly overstated, and the overstatement is expensive, so state the limit plainly. Quintile scores preserve relative rank within a stated population and window. They do not preserve absolute performance: the top monetary quintile exists even when spending has fallen for every customer in the base, and a customer whose score rises from 3 to 4 may have spent less than last year while others spent less still. Scores computed on different windows or different populations are therefore not comparable unless the changing cut points are reported alongside them — which makes the cut points part of the deliverable, exactly as Section 3.5.1 would predict.

DEFINITION

Quintile Scoring

Quintile scoring converts a numeric quantity into a five-level ordinal score by ranking all customers on the quantity and assigning 5 to the top fifth through 1 to the bottom fifth, with the direction chosen so that 5 is always the marketing-favorable end. Scores are relative to the scored population and window: they preserve rank, not level, and are not comparable across periods or populations unless the cut points are reported with them. The handling of tied values is declared as part of the scoring rule.

Source: Adapted from Hughes (1994).

Ties deserve the paragraph most treatments skip, because on real marketing data they are not an edge case. Frequency at StyleCraft is lumpy — a large share of customers have exactly one order, a designed fact of the New/At-Risk population — and five equal-count bins cannot be cut through a value that thousands of customers

share. Software resolves this either by complaining (pandas' qcut raises an error on duplicate bin edges) or by an arbitrary-but-deterministic rule. The error is the more useful outcome, because it is the data telling you the quantity has too few distinct values to support five honest bins.

Four repairs are available, and choosing among them is a scoring-rule decision that belongs in the written definitions. Use fewer frequency bands, with cut points at values customers actually take. Keep tied values together and accept unequal group sizes, by ranking densely or cutting on declared thresholds. Break ties on a second, genuinely behavioral quantity, so that the distinction means something. Or rank with a row-order tiebreaker — pandas' rank(method="first") — and label it, in the deliverable, as an arbitrary classroom convention. The last option is the one AI assistants reach for by default, and it is the weakest: it assigns different scores to customers with identical frequency on the basis of where they happen to sit in the file, so it is reproducible only while the sort order is, and the distinction it manufactures has no behavioral content whatsoever.

This guide takes the first repair. StyleCraft's operating frequency rule uses three declared bands rather than five quintiles: exactly one order in the window scores F1, two or three orders score F3, and four or more orders score F5. The bands are cut at values the distribution actually contains, every customer with the same order count receives the same score, and the mapping onto the five-point grid axis is stated rather than implied. Recency and monetary value keep their quintiles, because both are near-continuous and support five bins without ties. Code 6.3 in Lab 6.1 implements the diagnosis and the repair together.

With three scores per customer, the classical presentation collapses to the two most operational dimensions — recency and frequency — and arranges them as a grid whose regions carry conventional names. Table 6.3 is this guide's operational grid, and its most important property is structural rather than nominal: it is a complete lookup over all twenty-five recency-by-frequency cells, so every scored customer falls in exactly one region and no customer falls in two. That property is not decoration. The opening case requires every purchasing customer to receive exactly one journey, and a published grid with gaps or overlaps lets two analysts apply the same rules and reach different assignments — which is a governance failure before it is an analytic one. The names are operational shorthand, not discoveries: “Champions” is a label for the recent-and-frequent corner declared by the scoring rule, which is what makes RFM an a priori method in Section 6.4's sense — the grid exists before any customer is scored, and the data's job is to fill it.

Table 6.3

The named RFM grid: a complete lookup over recency and frequency scores

Recency score	F1	F2	F3	F4	F5
R5 (most recent)	New Customers	Potential Loyalists	Loyal Customers	Champions	Champions
R4	Potential Loyalists	Potential Loyalists	Loyal Customers	Champions	Champions
R3	Needs Attention	Needs Attention	Loyal Customers	Loyal Customers	Loyal Customers
R2	Hibernating	Needs Attention	At Risk	At Risk	At Risk
R1 (least recent)	Hibernating	Hibernating	At Risk	At Risk	At Risk

Two notes complete the grid's operating definition. First, under StyleCraft's three-band frequency rule the scored population occupies only the F1, F3, and F5 columns; F2 and F4 stay empty. They are retained in the published grid because the grid is the general lookup — a population whose frequency distribution supports five bands maps onto it without amendment — and because a lookup with missing columns is the defect this table exists to eliminate. Second, region membership is determined by the pair of scores and nothing else: monetary value remains attached to every customer and returns as a sizing and prioritization variable in Section 6.11, but it does not move a customer between regions. Table 6.4 gives each region its reading and its conventional treatment logic — and treats that logic, per the Definition box above, as a hypothesis about response rather than an established response rule.

Table 6.4

The seven grid regions: reading and conventional treatment logic

Region	Cells	Reading	Conventional treatment logic (a hypothesis to test)
Champions	R4–R5 × F4–F5	Bought recently, buy often	Reward and early access; protect rather than discount
Loyal Customers	R5×F3, R4×F3, R3×F3–F5	Steady repeaters, or recent moderate repeaters	Nurture; cross-category offers
Potential Loyalists	R5×F2, R4×F1–F2	Recent, not yet habitual	Onboard toward the second and third purchase
New Customers	R5×F1	Just arrived, one order	Welcome journey; measure, do not judge
Needs Attention	R3×F1–F2, R2×F2	Middling on both, drifting	Low-cost re-engagement test before recency falls further
At Risk	R1–R2 × F3–F5	Were frequent, have gone quiet	Win-back triggers, tested incentives
Hibernating	R2×F1, R1×F1–F2	Long gone, rarely bought	Low-cost reactivation or suppression

Read against the opening case, the grid is already a serious candidate map: it is behavioral, rule-defined, exhaustive, reproducible in the database, and its regions come with treatment logic attached — several of its rows are recognizably journeys. Its limits are equally visible and motivate the rest of the chapter. The grid sees exactly three behaviors, and its regions are drawn on two of them; StyleCraft's designed structure involves channel, discount dependence, and loyalty enrollment, none of which R, F, and M can express — the Discount Trend Follower and the Urban Loyal Core can occupy the same grid cell while requiring opposite treatments, and discounting is precisely the lever on which the "reward, do not discount" logic would misfire against a discount-dependent customer. The grid's boundaries are declared, not estimated, so it cannot surprise you. Widening the behavioral lens requires more features and a method that can weigh all of them at once — which is the business of Sections 6.6 through 6.10.

6.6 Building the Customer-Grain Feature Table

Both of this chapter's methods consume the same raw material: a table with one row per customer and one column per behavioral quantity. StyleCraft does not ship behavior at that grain — behavior lives in transactions, at order-line grain — so the table must be built, and the build is a grain change in exactly Section 3.3's sense, executed with the groupby mechanics established in Chapter 4's lab. This guide calls the result the customer-grain feature table, and the companion repository ships a completed version, customer_features, as the answer key the labs reconcile against; the columns are features in Section 3.5's role vocabulary because their destiny is to be model inputs.

CONCEPT

The Customer-Grain Feature Table

The customer-grain feature table is the analytic asset that powers customer-level modeling: one row per customer in a declared population, one column per feature, every feature computed from certified data by a written derivation rule at a declared analysis date and window. Building it is a grain change (Section 3.3), so it carries grain-change obligations: declare the source grain and target grain, declare which rows the population admits and which it excludes, state each feature's aggregation rule, and reconcile the result — row counts, sums, and distinct counts — against the certified source before anything downstream consumes it. The table is an asset, not a byproduct: later chapters in Part II model from this same table, so an error built into it now propagates through the rest of the book.

Source: Course concept developed for this guide, informed by the grain discipline of Section 3.3.

Before any feature is derived, the population must be settled, because the opening case contains a constraint the naive build cannot satisfy. The CRM manager must account for every signed-up customer, and behavioral features are undefined for a customer who has never purchased: recency has no last order to count back from, average order value divides by zero, and every purchase-derived share has an empty numerator over an empty denominator. Two operating designs resolve this honestly. A full-base segmentation includes every signed-up customer and declares zero-purchase behavior explicitly — frequency 0, monetary 0, average order value missing rather than zero, a separate recency rule keyed to signup, and purchase-derived shares recorded as not applicable. A purchasing-customer segmentation restricts the clustered population to customers with at least one purchase in the window and routes signups with no purchase in observed history to a separate activation program that sits outside the clustering.

This guide takes the second design, and states it as a rule rather than as a filter buried in a query. The customer_features table contains customers with at least one transaction in the declared window; its row count is therefore the certified distinct-purchaser count, not the customers table's row count. Customers outside it are not discarded, and naming them precisely matters more than it looks. The complement of the clustered population is not “never purchased.” It is “no purchase in the analysis window,” and it holds two quite different groups: customers with no purchase anywhere in the observed history, and customers who bought before the window opened and have since gone quiet. With twenty-four months of history on file, even the first group is properly described as having no purchase in observed history rather than none ever — the file cannot see further back than it goes. Code 6.5 separates the three populations and reports all three counts.

The destinations differ accordingly, and they fit the opening case's constraint rather than straining it. Customers with no purchase in observed history belong to the always-on welcome and activation program StyleCraft already runs — business as usual, continuing unchanged, and consuming none of the four new fall slots. Customers who purchased only before the window are lapsed rather than new, and they are candidates for win-back treatment inside the fall program; if a later release shortens the window, they re-enter the clustered population and the population rule has to be restated with it. Stating the population this way keeps the opening case's promise — every customer has exactly one destination — without pretending that a customer with no purchases can be described by purchase behavior, and without quietly inventing a fifth journey.

Table 6.5 specifies the features the labs build, their derivation rules, and — a distinction the rest of the chapter depends on — their role in the segmentation. Clustering inputs are the behavioral features the algorithm will see. Profiling holdouts are attributes deliberately withheld from the algorithm and reserved for Section 6.10's checks. And one column, exposure_months, is neither: it is a denominator, computed so that other features can be expressed as rates, and it never enters the distance calculation itself.

Table 6.5

The customer_features build: derivation rules and roles

Feature	Derivation rule (analysis date June 30, 2026; 24-month window)	Source	Role
recency_days	Days from the customer’s most recent order_date to the analysis date	transactions	Clustering input (state)
frequency	Count of distinct order_id in the window	transactions	Clustering input (level)
monetary	Sum of line_revenue in the window	transactions	Clustering input (level)
exposure_months	Days from the later of signup_date and the window start to the analysis date, divided by 30.44, then floored at 1.0 month	customers	Denominator only — never clustered
orders_per_month	frequency ÷ exposure_months	derived	Clustering input (intensity)
revenue_per_month	monetary ÷ exposure_months	derived	Clustering input (intensity)

Feature	Derivation rule (analysis date June 30, 2026; 24-month window)	Source	Role
aov	monetary ÷ frequency	derived	Clustering input (shape)
discount_share	Revenue on lines with discount_pct > 0 ÷ monetary	transactions	Clustering input (shape)
store_share	Distinct orders with channel = Store ÷ frequency	transactions	Clustering input (shape)
tenure_days	Days from signup_date to the analysis date	customers	Profiling holdout
loyalty_tier	As stored (ordinal per Table 3.5)	customers	Profiling holdout
app_user	As stored	customers	Profiling holdout
email_opt_in	As stored	customers	Profiling holdout
home_metro	As stored	customers	Profiling holdout
acquisition_channel	As stored	customers	Profiling holdout
age_band	As stored; partly missing, per Table 6.1	customers	Profiling holdout

Three build decisions deserve their reasons stated, because each is a judgment call an AI assistant will make silently. The first is the population rule just settled, which must be written down and reflected in every reconciliation. The second is the ratio features: aov, discount_share, store_share, and the two per-month rates are ratios, so each carries the full four-element definition of Section 3.5.1 and each has an edge-case story — a one-order customer's aov equals that order's value; a customer with no discounted lines has discount_share exactly 0, which is data rather than missingness; and exposure_months carries a declared floor of one month, because a customer observed for nine days cannot support a stable monthly rate.

The third decision is the one this chapter cares about most, and the draft version of this section got it half right. Holding tenure_days out of the clustering inputs stops the algorithm from using signup date directly. It does not stop the algorithm from rediscovering it. Frequency and monetary value computed over a 24-month file are mechanically bounded by how long the customer has existed: a customer who signed up nine weeks ago cannot have accumulated two years of orders or two years of spend even if her purchasing intensity is identical to a two-year customer's. Lifetime totals in a fixed window are, in part, a measure of exposure wearing the costume of value — and a segmentation that clusters on them alone will separate customers by age and then let a manager read the separation as quality.

The repair is to give the algorithm both readings and let the profile distinguish them. The level features (frequency, monetary) say what the customer has done in total; the intensity features (orders_per_month, revenue_per_month) say what the customer does per month of exposure, with exposure defined as observed time in the window rather than calendar tenure. A short-tenured customer buying briskly and a long-tenured customer buying rarely can share a lifetime total and sit far apart on intensity, which is exactly the distinction a treatment map needs. Note the deliberate asymmetry: tenure is excluded as a clustering coordinate and simultaneously used as a denominator. Be exact about what that buys. Expressing behavior per month of exposure reduces the mechanical effect of tenure and puts it on screen beside the lifetime totals; it does not eliminate the effect from a specification that still contains those totals, and the per-month rates are themselves least stable for the customers observed shortest, which is why exposure carries a floor. Clustering on tenure directly, by contrast, is how the effect is rediscovered and mistaken for a finding.

Two consequences follow for the sections ahead. First, the fourth segment the labs recover is a lifecycle position rather than a quality verdict — it collects customers whose observed value is low chiefly because their observed time is short, together with longer-tenured customers who have genuinely gone quiet; Section 6.14 shows how to read it and how not to. Second, the input set now contains features that are mathematically dependent on one another — aov is monetary divided by frequency, and each per-month rate is a level feature divided by exposure. Standardization will equalize their scales, and Section 6.7 will show that equalizing scale is not the same as

removing redundancy. The response is not to guess but to measure: the feature-set sensitivity analysis of Lab 6.2 refits without aov, without the level features, and without the intensity features, and reports how far the assignments move.

The build closes the way every grain change in this guide closes, with reconciliation to known totals. The certified file's verification log (Chapter 4) fixed the total revenue, the distinct order count, and the distinct purchasing-customer count; the feature table must return them exactly — the sum of monetary equals certified revenue, the sum of frequency equals certified distinct orders, and the row count equals certified distinct purchasers. Lab 6.1 performs the build twice, first on a hand-checkable miniature and then at full scale, and its Verification Checks are these three equalities plus the population statement that explains the gap between the feature table's row count and the customers table's.

6.7 Distance, Similarity, and the Tyranny of Units

Post hoc segmentation needs what a priori segmentation never did: a mathematical answer to the question “which customers are alike?” Clustering methods group customers by similarity, and the standard move is to define similarity as the inverse of distance: represent each customer as a point whose coordinates are the feature values, and call two customers similar when the points are close. The workhorse distance is Euclidean — the straight-line distance of geometry, extended to as many dimensions as there are features (Everitt et al., 2011).

DEFINITION

Euclidean Distance

The Euclidean distance between two customers represented as feature vectors is the square root of the sum of squared differences across their features. It is the default distance of k-means clustering; its value depends on the units and spread of every feature that enters it, which is why feature scaling is part of the method rather than an optional refinement.

Source: Adapted from Everitt et al. (2011).

The definition's second sentence is where segmentation projects quietly die, so it earns a worked example at hand scale. Take three customers from Lab 6.1's miniature, described on three features — recency in days, frequency in orders, monetary in dollars. C002, the occasion shopper: recency 120, frequency 2, monetary 400.00. C005, the frequent urban regular: recency 3, frequency 3, monetary 127.00. C006, the lapsed one-time big basket: recency 198, frequency 1, monetary 242.00. Intuition and marketing logic say C002 resembles C006 — infrequent, high-basket, fading — far more than either resembles C005. The raw arithmetic agrees, but examine why. The C002-C006 distance is the square root of $78^2 + 1^2 + 158^2$, which is the square root of 31,049, about 176.2 — and 24,964 of the 31,049, roughly 80 percent, is contributed by the monetary difference alone. The C002-C005 distance is the square root of $117^2 + 1^2 + 273^2$, about 297.0, with monetary contributing 84 percent. Frequency, the feature Chapter 5 suggested separates StyleCraft's worlds most sharply, contributes one part in tens of thousands to both distances. The dollars are not more important; they are merely measured in bigger numbers. Recency in days spans a couple of hundred; monetary in dollars spans hundreds and, at full scale, thousands; frequency spans single digits. Unscaled Euclidean distance is, on this table, monetary value with a rounding error — the clustering would dutifully return spend bands and a manager would call them segments.

The remedy is standardization: convert every feature to a common scale before any distance is computed. The standard choice is the z-score — subtract the feature's mean, divide by its standard deviation — after which every feature has mean 0 and standard deviation 1, and a difference of 1.0 means “one standard deviation apart” regardless of the feature's native units (James et al., 2021). On the standardized miniature, frequency's voice returns: C005's three orders against C006's one order becomes a difference of about two standard deviations, fully comparable to the monetary gap, and the recovered geometry matches the marketing intuition instead of the payroll.

DEFINITION

Standardization (z-Score Scaling)

Standardization rescales a numeric feature by subtracting its mean and dividing by its standard deviation, giving every standardized feature mean 0 and standard deviation 1 so that distance computations weight features by their variation in standard-deviation units rather than by their native units. The scaling parameters are computed from a declared population, are part of the segmentation's written definition, and must be retained with the fitted model if new customers are ever to be assigned consistently (Section 6.10).

Source: Adapted from James et al. (2021) and Pedregosa et al. (2011).

CONCEPT

Scaling Is a Modeling Decision, Not Plumbing

Standardization looks like preprocessing and is actually a declaration of what "alike" means: it asserts that one standard deviation of recency should count the same as one standard deviation of discount dependence. That is usually the right default and it is never neutral — an analyst who wanted monetary value to matter more could legitimately weight it, provided the weighting is declared and defended. The indefensible version is the silent one: skipping scaling entirely, which does not avoid the weighting decision but makes it by accident, handing the definition of customer similarity to whichever feature happens to be denominated in the largest numbers. When auditing any clustering — your own, a colleague's, or an assistant's — the first question is Chapter 3's question in new clothing: what, exactly, was measured? Here: what, exactly, was the distance computed on?

Source: Course concept developed for this guide, informed by James et al. (2021).

Three boundary notes complete the toolkit at working level. Standardization applies to numeric features; categorical attributes like home_metro do not subtract and divide, which is the technical half of the reason Table 6.5 holds them out for profiling. Z-scores do not remove skew — Chapter 5 showed monetary value is strongly right-skewed, and it remains right-skewed after standardization, so a heavily skewed feature can still exert outsized pull through its extreme values, and standardization computed from the mean and standard deviation is itself sensitive to those extremes (Pedregosa et al., 2011). Working practice ranges from accepting this, to log-transforming skewed monetary features, to the rank-based scoring RFM already performs; the labs use standardization and let the stability checks of Section 6.10 report whether extreme customers are steering the solution.

The third note is the one most often missed, and Section 6.6 has already set it up. Standardization equalizes scale. It does nothing about redundancy. If two features carry the same information — or if one is a function of two others, as aov is of monetary and frequency — standardizing them does not collapse them into one voice; it gives each an equally loud voice, so the underlying quantity speaks several times in the distance calculation and is weighted accordingly. This is not automatically an error, and it does not automatically demand that a feature be removed. It demands that the analyst know it is happening and test it: fit the proposed specification, refit with the composite dropped, refit with a component dropped, and compare the assignments and the profiles. Feature selection is part of the definition of distance, which means it is part of the definition of "alike," which means it is a modeling decision that belongs in the written specification rather than in a default. Lab 6.2 runs exactly that comparison.

6.8 k-Means Clustering: Intuition, Algorithm, and Sensitivities

With customers as scaled points and similarity as distance, clustering is the task of partitioning the points into groups that are compact within and separated between. This guide's method is k-means, the most widely used

clustering algorithm in marketing practice — chosen here for the same reasons it is chosen in industry: it is fast at customer-base scale, its output is easy to profile, and its failure modes are well understood, which for a verification-centered course is a feature (MacQueen, 1967; Lloyd, 1982; James et al., 2021). It also belongs to a family the rest of Part II does not: k-means is unsupervised: there is no outcome variable, nothing is predicted, and the algorithm is given no answer to fit toward (Pedregosa et al., 2011). That is precisely why validation has to be constructed rather than measured.

The intuition first. Suppose the number of groups, k, is given — Section 6.9 takes up where it comes from. k-means seeks k cluster centers, called centroids, such that every customer is assigned to the nearest centroid and the total of squared distances from customers to their assigned centroids — a quantity called inertia, or within-cluster sum of squares — is as small as possible. A good solution is one where customers sit close to their centers: compact clusters, and a centroid that genuinely typifies its members, because each centroid is simply the average of its cluster on every feature. That last fact is the profiler's gift — a centroid is a readable customer sketch in standardized units — and it is also a hand-verifiable claim Section 6.12's audit routine will exploit.

DEFINITION

k-Means Clustering

k-means clustering partitions observations into a chosen number k of clusters by locating k centroids and assigning each observation to its nearest centroid so as to minimize inertia, the total within-cluster sum of squared distances. It is fitted by alternating assignment and centroid-update steps from an initial placement of centroids, and its solution can depend on that initialization. It is an unsupervised method: it uses no outcome variable and predicts nothing.

Source: Adapted from MacQueen (1967), Lloyd (1982), and Pedregosa et al. (2011).

The algorithm sketch is short enough to hold in your head, and holding it there is what demystifies the output. Step zero: place k centroids somewhere — classically at random customer positions. Step one, assignment: give every customer to its nearest centroid. Step two, update: move each centroid to the mean of the customers just assigned to it. Repeat steps one and two; each round can only lower inertia, so the alternation settles — converges — when assignments stop changing. The procedure is a loop a student can trace on paper with a dozen points, and Lab 6.2's first Verification Check does exactly that in miniature.

Convergence, however, is not correctness, and the algorithm's two sensitivities are the chapter's verification theme in mathematical form. The first is initialization sensitivity: the loop settles into a locally best solution reachable from wherever the centroids started, and different random starts can settle into different partitions with different inertias (Pedregosa et al., 2011). Practice answers with better starts and more of them — the k-means++ initialization spreads the initial centroids apart, and standard implementations rerun the whole procedure from multiple starts, keeping the lowest-inertia result (Arthur & Vassilvitskii, 2007) — but “the software reruns internally” does not discharge the analyst's obligation: a solution worth deploying should be demonstrated stable across independently seeded fits, which is Section 6.10's stability check. The second sensitivity is the one the opening case turns on, and it deserves its own box.

CONCEPT

The Algorithm Returns the k You Asked For

k-means answers the question it is asked, which is not the question the marketer is asking. Asked for nine clusters, it partitions the customers into nine — optimally arranged, confidently bounded — whether the data contain nine groups, four, or none at all; a continuous but unstructured customer cloud can be sliced into k tidy regions with the same serene output. (The one exception is degenerate data with fewer distinct points than requested clusters, where an implementation may return fewer clusters and warn.) The algorithm cannot report "there are no segments here," and it will never volunteer that a different k fits better. Every claim of the form "the analysis found nine segments" is therefore, uninspected, a claim about the prompt, not about the customers. What converts a partition into evidence of structure is everything around the algorithm: the choice of k defended on multiple instruments (Section 6.9), stability across refits and samples, and separation on variables the algorithm never saw (Section 6.10). The intern's overnight notebook is not wrong; it is unexamined — and this box is the reason the distinction matters.

Source: Course concept developed for this guide, informed by Pedregosa et al. (2011).

Two further properties round out the working-level picture, both consequences of the squared-Euclidean objective. k-means prefers compact, roughly round clusters of broadly similar spread; genuinely elongated, unevenly sized, or nested structures get carved into pieces, which at customer-base resolution is usually tolerable and occasionally the finding (Pedregosa et al., 2011). And because means are sensitive to extremes, a handful of outlying customers — Chapter 5's top-decile whales — can drag a centroid toward themselves; the stability checks and the concentration awareness of Section 5.9 are the standing defenses. Neither property demands a more exotic algorithm for this course's purposes; both demand the profiling and validation that the next two sections install.

6.9 Choosing k: Elbow, Silhouette, and Business Judgment

Post hoc segmentation's defining freedom — that group membership is estimated rather than declared — arrives with a question the algorithm cannot answer for itself: what should k be? The honest answer is that k is chosen, not discovered, and it is chosen with instruments plus judgment. This section supplies the two standard instruments and then insists on the third leg, because on real customer data the instruments alone are routinely indecisive and occasionally misleading.

The first instrument reads inertia. Fit k-means across a range of candidate k values — the labs use 2 through 8 — and plot each solution's inertia. At the optimum, inertia is non-increasing as k rises (more centers, shorter distances; at the absurd limit of one cluster per customer it reaches zero), so the plot's level is uninformative; its shape is the signal. Where adding a cluster stops buying much compactness, the curve bends, and the bend — the elbow — marks the k beyond which additional clusters are subdividing structure rather than finding it (Thorndike, 1953).

DEFINITION

Elbow Method

The elbow method selects candidate values of k by plotting inertia against k and looking for the bend where further increases in k yield sharply diminishing reductions in inertia, on the reasoning that k values past the bend split real groups rather than separate them. Its verdicts are visual and frequently ambiguous, so it nominates candidates rather than deciding.

Source: Adapted from Thorndike (1953).

Two properties of that curve are worth fixing before Lab 6.2 reads one. At the optimum inertia is non-increasing rather than strictly decreasing: an added centroid that buys no improvement leaves it flat, and a flat step is information rather than a defect. And because each candidate k is a separately initialized optimization, a higher-k fit can settle into a worse local minimum than a lower-k fit and report a small rise. A rise is therefore a warning to inspect initialization and convergence and to refit with more starts — not, by itself, proof that the code is broken.

The second instrument reads separation per customer. The silhouette score asks, for each customer, how close it sits to its own cluster compared with the nearest other cluster, scaled to lie between −1 and +1: near +1 means snugly placed, near 0 means on a boundary, negative means arguably misassigned (Rousseeuw, 1987). Averaged over all customers, the silhouette gives each candidate k a single separation grade, and the k with the highest average is the instrument's nominee. The silhouette is more decisive than the elbow and correspondingly more dangerous to obey blindly: on customer bases with one dominant divide — StyleCraft's urban-suburban split is exactly such a divide — the silhouette often crowns k = 2, because the coarsest cut is the cleanest, even when finer structure is real, designed, and strategically the entire point.

DEFINITION

Silhouette Score

A customer's silhouette compares its average distance to members of its own cluster with its average distance to members of the nearest other cluster, yielding a value between −1 and +1 where higher means better placed. The mean silhouette across customers summarizes a clustering's separation and is compared across candidate values of k. It grades geometry only: a high silhouette says the partition is clean, not that its groups are marketing segments.

Source: Adapted from Rousseeuw (1987).

The third leg is business judgment, and calling it a leg rather than a tiebreaker is deliberate: the instruments measure geometry, and geometry is not the objective — a segmentation is chosen for treatment, and Section 6.2's treatment question bears directly on k. The platform sustains four journeys; a k of 9 is undeployable this fall regardless of its silhouette, and a k of 2 discards distinctions — discount dependence, lifecycle stage — that the treatments exist to exploit. Judgment also carries domain knowledge the geometry cannot see: Chapter 5's cohort and concentration evidence, the designed logic of the expansion story, and the practical requirement that every segment be large enough to justify a journey (Section 6.11's substantial criterion, previewed). Table 6.6 summarizes the three legs and, importantly, what each cannot do.

Table 6.6

The three legs of choosing k

Leg	What it measures	What it cannot do
Elbow (inertia)	Diminishing returns in compactness as k grows	The bend is often gradual; nominates a range, rarely a value
Silhouette	Average quality of each customer’s geometric placement	Favors the coarsest clean cut; blind to strategy and treatment capacity
Business judgment	Deployability, treatment capacity, domain structure, segment sizes	Cannot certify that the preferred k matches structure — that is the instruments’ and the validation checks’ job

The working procedure, which Lab 6.2 executes, braids the legs rather than ranking them: compute elbow and silhouette across the candidate range; shortlist the k values that either instrument nominates; profile the solutions on the shortlist (Section 6.10); and choose on treatment grounds among solutions the instruments and profiles both support — documenting the choice and the rejected candidates, because “why four?” is the first question the sign-off meeting will ask, and “the silhouette said so” is not an answer that survives this section's own caveats. On StyleCraft

the braid has a designed resolution: the instruments shortlist a small range, the profiles at k = 4 align with recognizable and treatable customer stories, and — uniquely to a designed dataset — the answer key confirms it. The documentation habit, not the confirmation, is what transfers to data without keys.

6.10 Profiling, Validating, and Naming Segments

A converged, k-chosen clustering is still only a partition. This section converts it into a segmentation the CRM manager can sign: first by profiling — describing each cluster honestly enough to see what it is — then by validating — testing what the partition can and cannot be shown to reflect — and finally by naming, ruling, and preserving — translating clusters into words, into written assignment rules, and into a saved model the organization can actually run. All three steps consume Chapter 5's toolkit; what this chapter owns is their application to algorithmic groups and the two checks, stability and holdout profiling, that algorithmic groups uniquely require.

Cluster profiling is the customer profile of Section 5.6 with clusters as the grouping variable, and every discipline transfers: sizes first, centers with spreads, declared grains, distributions shown where shapes are mixed. The centroid table is the natural spine — k rows, one per cluster, features as columns, values as within-cluster means — read first in standardized units (which features define each cluster: “recency 1.4 standard deviations better than average”) and then translated back to native units for the deck (“median 11 days since last purchase”), because standardized units are for analysts and days and dollars are for decisions. Two Chapter 5 cautions bite with extra force here. Cluster means are means: a cluster's monetary centroid can be propped by a few whales, so spreads and medians travel with it. And cluster sizes are findings, not footnotes: size is the first row of the profile and the first input to the substantiality judgment of Section 6.11.

Validation is the new obligation, and it has two instruments that answer two different questions. The first is the stability check, and its question is whether the partition is a property of the data or of the fitting run: a segmentation worth deploying should not depend on the random seed, the particular sample, or the day the notebook ran. The operational tests are refits under perturbation — refit from several independent seeds, refit on random subsamples of, say, 80 percent of customers — followed by a comparison of the assignments. The comparison is where practitioner treatments usually go wrong. Cluster labels are arbitrary from run to run, so the naive procedure — cross-tabulate and count the customers outside each row's dominant cell — quietly assumes the runs line up one to one, and they need not: two clusters in the baseline can both have their largest overlap with the same cluster in the refit, and the arithmetic will report stability that does not exist.

Use a measure built for the problem instead. The adjusted Rand index compares two partitions by asking, over all pairs of customers, how often the two solutions agree about whether the pair belongs together — a quantity that is invariant to relabeling and corrected for the agreement expected by chance (Hubert & Arabie, 1985). It reaches 1.0 for identical partitions and sits near 0.0 for unrelated ones, and because the chance correction can be exceeded downward it also goes negative, to a lower bound of −0.5 for especially discordant partitions; a negative value means the two solutions agree less than chance would predict (Pedregosa et al., 2011). Where a customer-level reassignment rate is genuinely wanted for the deck, align the labels first with an explicit one-to-one assignment, then count. Precision of language matters here too: a scikit-learn fit with n_init = 10 is already the best of ten internal starts, so five seeded fits are “five independently seeded fits, each retaining the best of ten initializations,” not “five random starting points.”

DEFINITION

Stability Check

A stability check assesses whether a clustering solution reflects structure in the data rather than an accident of fitting, by refitting under perturbation — independently seeded initializations, random subsamples — and measuring agreement between the resulting partitions with a relabeling-invariant index such as the adjusted Rand index such as the adjusted Rand index, which is 1.0 for identical partitions, near 0.0 for unrelated ones, and negative for partitions agreeing less than chance would predict. Low agreement indicates a partition the data do not support at the chosen k. High agreement establishes that the partition is reproducible; it does not establish that the groups are meaningful.

Source: Adapted from Everitt et al. (2011); index per Hubert and Arabie (1985).

The second instrument is holdout profiling, and it answers a different question: not "is this partition reproducible?" but "does it line up with anything outside its own inputs?" Recall Table 6.5's division: the algorithm clustered on behavior only, while metro, loyalty tier, app use, acquisition channel, age band, and tenure were withheld. Now profile the clusters on the withheld attributes. If a cluster built purely from purchasing rhythm turns out to be overwhelmingly app-using, loyalty-enrolled, and urban — attributes the algorithm never saw — the clusters are at least describing customers who differ in ways beyond the features that made them. Holdout profiling is predict-then-verify (Section 1.7) at the level of an entire analysis: before running the holdout profile, the analyst writes down what each cluster should look like on the withheld attributes if it is the kind of customer the behavioral profile suggests — and then checks.

Be exact about what this establishes, because the temptation to overclaim here is strong and the overclaim is the kind a sign-off meeting repeats. The withheld attributes are not statistically independent of the clustering inputs — metro, app use, loyalty enrollment, acquisition channel, and purchasing behavior are correlated with one another — so agreement between them is convergent, interpretive evidence rather than independent proof. It raises confidence that the clusters are interpretable and correspond to describable kinds of customers; it does not demonstrate that they are natural types, and it emphatically does not demonstrate that they will respond differently to treatment, which is a causal claim only a test can support. The converse is equally worth stating: a behaviorally useful segmentation may show little demographic separation and remain entirely valid for CRM purposes. Failed predictions are findings rather than embarrassments — a "loyal regulars" cluster that turns out to be half discount-dependent has just told you the behavioral features were incomplete, one section before the naming step would have shipped the error.

CONCEPT

Check Clusters on Variables They Never Saw

Withhold meaningful attributes from the clustering inputs; predict, from each cluster's behavioral profile, how it should differ on the withheld attributes; then profile and compare. Confirmed predictions are one layer of external-variable evidence that no inertia curve or silhouette can supply. They are not proof of natural kinds: the holdout attributes are correlated with the inputs, so agreement is convergent rather than independent, and nothing in the exercise speaks to how the groups will respond to being treated differently. Report holdout agreement as what it is — supporting evidence for interpretability — and leave response claims to the experiments of later chapters.

Source: Course concept developed for this guide, informed by Everitt et al. (2011) and Pedregosa et al. (2011).

Naming comes next, deliberately late, because names are the most contagious artifact a segmentation produces: the profile stays in the appendix, the name goes in the calendar, the deck, and eventually the org chart. Two rules keep names honest. Name from evidence: every word in a segment name must be redeemable against the profile — a cluster may be called "Store-Only Occasion Shoppers" if store_share and aov say so, and may not be called

“Affluent” on any evidence StyleCraft's schema contains, which retires the intern's assistant's christening and enforces Section 6.3's base-vocabulary discipline. And name the behavior, not the person, where possible: behavioral names (“Discount-Dependent,” “One-Purchase Recent”) describe what the data measured; person names (“Bargain Hunters”) drift toward the psychographic claims the data cannot support and invite the gap-instinct misreadings Section 5.14 warned about.

The last step is the one most chapters omit, and omitting it is how segmentations die in production. A clustering is not a set of thresholds; it is a fitted object. Assigning a customer to a cluster requires the feature definitions, the scaler's means and standard deviations, the fitted centroids, and a nearest-centroid rule applied in standardized space. Simplified business rules — a WHERE clause on two or three columns — are useful and often necessary, because campaign platforms speak SQL rather than Python, but a threshold rule written by eye from a centroid table does not reproduce the fitted assignments and should never be assumed to. The professional practice is to do both and reconcile: retain and version the scaler, the model, and the feature definitions as the assignment system of record; publish the simplified rule as the operational approximation; and report the agreement between them, naming the customers on which they disagree. Discovery was statistical; deployment is governance, and governance runs on artifacts that can be rerun.

CONCEPT

Preserve the Model, Not Just the Rules

A deployed segmentation needs five things retained together and versioned as one release: the feature definitions and the analysis window; the fitted scaler's means and standard deviations; the fitted centroids; the nearest-centroid assignment procedure in standardized space; and the date and population on which all of it was fitted. New customers are then assigned by transforming their features with the saved scaler and predicting with the saved model — not by re-fitting, which would silently redraw the segments, and not by eyeballed thresholds, which approximate the model without reproducing it. Where a simplified rule is published for a campaign platform, measure its agreement with the model's assignments and document the exceptions. A segmentation nobody can rerun is a slide, not a system.

Source: Course concept developed for this guide, informed by Pedregosa et al. (2011).

6.11 From Segments to Targets: Useful-Segment Criteria and Segment Attractiveness

Verified, profiled, named segments still leave the strategic decision standing: which segments get which of the four journeys — and does every segment deserve one at all? This section supplies the two evaluation frames marketing strategy uses: the usefulness screen, which every segment must pass to be operable, and the attractiveness comparison, which ranks the segments that pass.

The usefulness screen is the classical five-criterion test (Kotler & Keller, 2016). A segment is measurable if its size and characteristics can be computed from available data; substantial if it is large and valuable enough to justify differentiated treatment; accessible if the firm can actually reach it through channels it operates; differentiable if it responds differently to marketing than other segments; and actionable if the firm can design and execute distinct treatment for it within real constraints.

DEFINITION

Useful-Segment Criteria

The useful-segment criteria hold that a segment justifies differentiated marketing only if it is measurable (size and traits computable from data), substantial (large and valuable enough to serve profitably), accessible (reachable through operable channels), differentiable (distinct in its response to marketing), and actionable (treatable distinctly within the firm's actual capabilities).

Source: Adapted from Kotler and Keller (2016).

Two of the five criteria are routinely mis-scored in practice, and both errors are worth naming before the screen is applied. The first is differentiability. Clustering establishes that groups differ on the features that made them, and profiling establishes that they differ on some attributes that did not. Neither establishes differential response, which is a statement about what happens when treatment changes and therefore a causal claim. A descriptive profile nominates a treatment hypothesis; a campaign or an experiment validates it. Scored honestly, the differentiable cell of a fresh segmentation reads “distinct behavioral profile; differential response untested,” and the treatment map's first year is partly an apparatus for testing it.

The second is substantiality, where the usual shorthand — a small segment is a curiosity — is too categorical to be useful. A segment of ninety customers may be exceptionally valuable per customer, strategically important as a beachhead in a new market, cheap to serve through an existing channel, legally or reputationally consequential, or valuable as a test cell that de-risks a larger rollout. What size actually determines is the burden of proof: a small segment must justify its journey on value, growth, strategic importance, and treatment economics rather than on headcount, and the smaller it is, the more of that case has to be made explicitly. Substantiality is a threshold question about whether differentiated execution pays, not a headcount test.

The criteria do their best work applied concretely, and Table 6.7 applies them to the four segments the labs recover — profiled per Section 6.10 and matching, by design, the dataset's planted structure. Reading the table row by row is a rehearsal for the sign-off meeting, and three cells deserve narration. Accessibility is where the suburban segment's weakness concentrates: a store-only, low-opt-in, no-app population is expensive to reach with a CRM program that lives in email and app — a finding that reframes “high AOV” considerably. Actionability is where the platform constraint rules: four journeys is the actionability budget, and it is why the agency's seven personas and the intern's nine clusters fail this screen before any statistics are consulted. And the New/At-Risk row carries the chapter's standing caveat in two cells at once — the segment is perfectly measurable, but what it measures is lifecycle position as much as value, and its response is not merely undifferentiated but unobserved.

Table 6.7

The usefulness screen applied to the four recovered segments

Criterion	Urban Loyal Core	Suburban Occasion	Discount-Dependent	New / At-Risk
Measurable	Yes — fitted behavioral assignment procedure	Yes — fitted behavioral assignment procedure	Yes — fitted behavioral assignment procedure	Yes — but measures lifecycle position as much as value (Section 6.14)
Substantial	Large share of customers and revenue	Smaller count; high revenue per order; margin caveat	Meaningful count; thin margin	Large count; low observed value largely by construction
Accessible	High — email and app reachability, with strong loyalty enrollment	Low — store-only, weak opt-in, no app	High — reachable on promotional channels	Moderate — recent signups, opt-in fresh
Differentiable	Distinct behavioral profile; differential response untested	Distinct rhythm and basket; response untested	Purchases concentrate on discounted lines; response to discount and brand treatments untested	Profile confounded with exposure; response unobserved
Actionable	Yes — reward and early-access journey	Partially — occasion triggers possible; channel limits bite	Yes, carefully — Section 6.16's trap lives here	Only after a tenure-based branch separates recent signups from established customers who have lapsed; the branch sits inside the segment's one journey slot and is declared as part of the assignment rule

Applied strictly, that qualification means none of the four segments has yet passed all five criteria, and the chapter will not pretend otherwise. Among segments that provisionally pass the measurable, substantial, accessible, and actionable portions of the screen, attractiveness analysis supports an initial targeting decision while differential response remains a treatment hypothesis to test. Attractiveness is that decision's evaluation frame: segments are compared on current size and value, growth trajectory, margin quality, reachability cost, strategic fit with the firm's direction, and fit with the firm's capabilities (Kotler & Keller, 2016; Wedel & Kamakura, 2000). The analyst's contribution is to put evidence in every cell of that comparison — sizes and values from the profile, growth from Chapter 5's cohort tables, margin from the unit-cost story, reach cost from the accessibility facts — and then to mark, honestly, which cells are evidence and which are judgment.

DEFINITION

Targeting and Segment Attractiveness

Targeting is the selection of which segments to serve and with what priority and resources. Segment attractiveness is the comparative evaluation that informs it, weighing each qualifying segment's size, value, growth, margin, cost to reach, and fit with the firm's strategy and capabilities — an evidence-plus-judgment ranking, not a computation. Cells sourced from the analysis and cells sourced from management judgment are marked separately, because the two carry different warranties.

Source: Adapted from Kotler and Keller (2016) and Wedel and Kamakura (2000).

Table 6.8 gives the frame a worked shape, and building it is the deliverable that connects the clustering to the four slots. Its rows are the qualifying segments; its columns run from the arithmetic the profile supplies to the judgment the room supplies, in that order, with an explicit evidence-versus-judgment declaration so that no reader

has to guess which is which. Code 6.15 produces the profile-arithmetic rows directly. The same-age trajectory row is evidence rather than judgment but it comes from elsewhere — it is merged in from the Chapter 5 cohort table, because a cohort view needs a period axis the feature table does not carry. The remaining rows, cost to serve and strategic fit and the declaration and the recommended priority, are written by the analyst and defended in the meeting.

Table 6.8

The segment-attractiveness matrix (structure; values computed in Lab 6.2)

Dimension (source)	Urban Loyal Core	Suburban Occasion	Discount-Dependent	New / At-Risk
Customers, and share of base (profile)	—	—	—	—
Revenue in window, and share (profile)	—	—	—	—
Revenue per customer (profile)	—	—	—	—
Revenue per exposure-month (profile)	—	—	—	—
Same-age trajectory (Chapter 5 cohort view)	—	—	—	—
Discount share of revenue, a margin proxy (profile)	—	—	—	—
Reachability: share app or email reachable (holdout profile)	—	—	—	—
Cost to serve and to treat (judgment, with finance)	—	—	—	—
Strategic fit with the expansion (judgment)	—	—	—	—
Evidence versus judgment (declared)	—	—	—	—
Recommended priority and journey (decision)	—	—	—	—

Attractiveness analysis is where descriptive language most wants to become causal, and the insight-statement discipline of Section 5.9 patrols the line. The data support “the Urban Loyal Core holds the largest share of repeat revenue and is the cheapest segment to reach”; they do not yet support any claim about what targeting it will cause. Every recommended priority in the last row of Table 6.8 is a decision made under that limitation, which is an argument for making the limitation visible rather than for postponing the decision.

The section's output, and the chapter's strategic deliverable, is the treatment map: each retained segment paired with a journey, a priority, and the one-line evidentiary basis; each excluded grouping paired with the criterion it failed; and the customers with no purchase in the window split between the always-on activation program and the win-back treatment, per Section 6.6. That map is what four slots on the fall calendar actually purchase — and producing it defensibly required every section behind it.

6.12 AI as a Segmentation Assistant

Segmentation is the point in this guide where AI assistance changes character. In Chapters 4 and 5, the assistant drafted code whose outputs the analyst could check against arithmetic — totals, counts, recomputed cells. In this chapter the assistant can draft the entire analytic act: prompted with a customer file, a current assistant will build features, scale them, choose a k, fit the clustering, name the segments, and write the strategy narrative, delivering in one response what the chapter just spent nine sections disciplining. The intern's overnight notebook is not a hypothetical; it is the new default state of the world. The division of labor therefore has to be stated with more care here than anywhere in Part I.

Where the assistant genuinely helps, use it, and the list is substantial. Drafting the feature-table build — the groupby scaffolds, the ratio features with their edge cases, the reconciliation printout — is legitimate delegation on mechanics Chapter 4 established, provided the derivation rules come from your written definitions rather than the assistant's defaults. Generating the evaluation loop — fit across k = 2 through 8, collect inertias and silhouettes, plot both — is boilerplate the assistant writes faster than you do. Proposing candidate features is a useful brainstorm ("what behavioral ratios might separate promo-driven customers?") provided proposals enter the specification as candidates, not conclusions. And drafting profile narration from a finished, verified centroid table can accelerate the deck, provided every sentence survives Section 6.10's naming rules and Section 5.9's insight-statement discipline.

What must never be delegated are the judgments, and this chapter's failure modes are characteristic enough to name individually. The confident k: asked to "segment these customers," assistants pick a k — often whatever a single silhouette sweep nominates, often simply a plausible-sounding number — and present it as found rather than chosen, with none of Section 6.9's braid. The skipped scaling: assistants sometimes cluster raw features, and Section 6.7 showed exactly what that produces — spend bands in segment costume; the error is invisible in the output, which is what makes it dangerous. The invented persona: asked to name or describe clusters, assistants reliably enrich them with attributes no input contained — incomes, life stages, attitudes — because fluent persona language saturates their training; "Affluent Suburban Loyalists" was generated by exactly this reflex, asserting a base the schema lacks (Section 6.3) and a loyalty relationship the holdout profile contradicts. The single-run solution: assistants fit once and report; nothing in a default workflow refits, perturbs, or compares partitions, so initialization luck ships as structure. The arbitrary tie rule: asked to score RFM quintiles on lumpy frequency, assistants reach for a row-order tiebreaker and never mention that the resulting distinctions have no behavioral content (Section 6.5). The narrated geometry: asked to interpret clusters, assistants convert arithmetic into story — "this segment is disengaging because…" — attaching motives and causes to what is, so far, a partition of a feature table. And the overfit taxonomy: more clusters always fit the fitted file more snugly, and an assistant asked for "detailed segments" will happily oblige with nine, geometry's version of the actionability failure in Section 6.11.

AI IN PRACTICE

The Cluster Solution That Must Survive a Refit

A productive pattern for delegated segmentation, in two prompts and an audit. Prompt one, the specification: paste the feature list with derivation rules (Table 6.5), the population rule, the analysis date, the certified totals, and the platform's treatment capacity, then ask for the build and the evaluation sweep only — "Build customer_features per these rules; print the three reconciliation figures; fit k-means for k = 2 through 8 on standardized inputs with a fixed random seed; report inertias and mean silhouettes; recommend nothing yet."

Prompt two, after you have chosen k per Section 6.9's braid: request the fit, the centroid table in standardized and native units, and the validation artifacts — adjusted Rand indices across five independently seeded fits and two 80 percent subsamples, and the holdout profile on the withheld attributes. Require that the fitted scaler and model be returned as objects, not as prose thresholds.

Then audit in four moves before any name or narrative is written: verify one centroid by hand (filter the feature table to one cluster, average one feature, compare); read the stability indices and reject the solution if the partitions do not reproduce; check the holdout profile against the predictions you wrote down before running it; and strike every noun in every segment name that no input feature or holdout column can redeem. Predictions first, per Section 1.7's predict-then-verify; every exchange documented per Appendix D — the assistant fitted the clustering, but the analyst of record certified that it is structure and not luck.

The deeper point extends the guide's standing division of labor to its first genuinely algorithmic output. In Chapter 5 the assistant drafted summaries and the analyst audited the claims; here the assistant drafts a model of the

customer base and the analyst audits the model — scaling, k, stability, holdout agreement, and every noun in every name. Appendix C provides prompt templates for the feature build and the audit routine; Appendix D's documentation requirement applies to every exchange; and the labs that follow are built so the audit has teeth, because on StyleCraft's designed data the truth is known: four segments were planted, and an analysis that cannot recover them — or that confidently reports nine — has failed in a way the answer key will name.

6.13 Hands-On Application in Python and Google Colab

The preceding sections built the method; this section runs it against the opening case, in two labs that mirror the chapter's two halves. Lab 6.1 builds the raw material: the customer-grain feature table, first on a ten-customer miniature where every number can be verified by hand, then at full scale with reconciliation to certified totals, closing with the RFM grid. Lab 6.2 performs the discovery: scaling, k-means, the choice of k, profiling, and the validation checks — ending with the comparison this dataset uniquely permits, recovered clusters against the four designed segments, and with the two artifacts the CRM manager actually deploys: the attractiveness matrix and the saved assignment model. Throughout, follow the course division of labor: AI assistants may draft the code (Appendix C has prompting templates; Appendix A covers Colab mechanics), but every output is predicted before it is computed, every build is reconciled to certified totals, and every AI exchange is documented per Appendix D.

6.13.1 Lab 6.1, Part A: The Feature Table in Miniature

The miniature is a twenty-order extract for ten StyleCraft customers, constructed in the notebook at order grain — Chapter 5's labs already practiced the line-to-order roll-up, so this lab starts one step later and points there rather than repeating it. The analysis date is June 30, 2026, matching Lab 3.1.

Code 6.1. Create the miniature order extract

```
import numpy as np
import pandas as pd

analysis_date = pd.Timestamp("2026-06-30")

orders_mini = pd.DataFrame({
    "customer_id": [
        "C001","C001","C001","C001","C002","C002","C003","C003",
        "C003","C004","C005","C005","C005","C006","C007","C008",
        "C008","C009","C010","C010"],
    "order_id": [
        "O01","O02","O03","O04","O05","O06","O07","O08","O09",
        "O10","O11","O12","O13","O14","O15","O16","O17","O18",
        "O19","O20"],
    "order_date": pd.to_datetime([
        "2026-06-21","2026-05-30","2026-04-18","2026-02-07",
        "2026-03-02","2025-11-19","2026-06-05","2026-04-24",
        "2026-01-16","2026-06-12","2026-06-27","2026-06-10",
        "2026-05-02","2025-12-14","2026-02-27","2026-06-18",
        "2026-03-29","2026-06-24","2026-05-16","2026-01-03"]),
    "order_revenue": [
        58.00, 47.00, 39.00, 52.00, 214.00, 186.00, 21.60,
        28.80, 19.20, 44.00, 36.00, 62.00, 29.00, 242.00,
        24.00, 71.00, 55.00, 38.00, 167.00, 145.00]
})

# The miniature's certified totals, written as checks rather than prose.
assert len(orders_mini) == 20
assert orders_mini["order_id"].is_unique
assert orders_mini["customer_id"].nunique() == 10
assert np.isclose(orders_mini["order_revenue"].sum(), 1578.60)
assert (orders_mini["order_date"] <= analysis_date).all()
```

```
print(len(orders_mini), "orders |",
      orders_mini["order_revenue"].sum().round(2), "total revenue")
```

Expected output: 20 orders and a total of 1578.6, with five assertions passing silently.
Input: twenty literal orders for ten customers. Transformation: none yet. Output: the extract's certified totals, recorded now because every build below must return them. The assertions are the point of the cell: Chapter 4 argued that a prediction belongs in the log before the step runs, and an assertion is that prediction written where the notebook itself will enforce it. The last one guards the recency calculation that follows, since an order dated after the analysis date would produce a negative recency — a Chapter 4-grade date defect rather than a customer behavior. The grain change is one grouped aggregation, exactly Chapter 4's mechanics with this chapter's derivation rules:

Code 6.2. Build and reconcile the miniature feature table

```
cf = (orders_mini
      .groupby("customer_id")
      .agg(last_order=("order_date", "max"),
           frequency=("order_id", "nunique"),
           monetary=("order_revenue", "sum"))
      .assign(recency_days=lambda d: (analysis_date
                                      - d["last_order"]).dt.days,
              aov=lambda d: (d["monetary"] / d["frequency"]).round(2))
      .drop(columns="last_order")
      .round(2))

# The grain change must preserve the extract's certified totals, and the
# one-order edge case must behave the way the definition says it will.
assert len(cf) == 10
assert cf["frequency"].sum() == 20
assert np.isclose(cf["monetary"].sum(), 1578.60)
assert (cf["recency_days"] >= 0).all()
assert np.allclose(cf.loc[cf["frequency"] == 1, "aov"],
                   cf.loc[cf["frequency"] == 1, "monetary"])

print(cf.sort_values("recency_days"))
```

Expected output: ten rows sorted from C005 at 3 days to C006 at 198 days; C002 with frequency 2, monetary 400.00, aov 200.00; C006 with frequency 1 and monetary equal to aov at 242.00; and five assertions passing.
Input: the twenty certified orders. Transformation: rows are collected by customer, and two features are derived from the aggregates. Output: a ten-row customer-grain table. The three reconciliation assertions are the grain-change identities of Section 6.6 in executable form — row count is the distinct-customer count, frequency total is the order count, monetary total is the extract's certified revenue — and if any of them fails, the build has silently dropped or double-counted something. Finding that on ten customers is the entire reason the miniature exists.

VERIFICATION CHECK

Before you run: compute two rows entirely by hand from Code 6.1's extract. C002's row: last order March 2, 2026, so recency_days = 120 (29 remaining March days + 30 + 31 + 30); frequency = 2; monetary = 400.00; aov = 200.00. C006's row: recency_days = 198, frequency = 1, monetary = aov = 242.00. Then predict the two structural facts the assertions encode — ten rows, twenty orders, 1578.60 in revenue — and predict that C005 is the most recent customer at 3 days.

After you run: all five assertions pass silently and the printed table matches both hand-computed rows and the sort order.

> Investigate if: a reconciliation assertion raises. A failed row count means the grouping key contained a missing value; a failed frequency total means orders were counted at line grain somewhere upstream; a failed monetary total means a filter ran that nobody declared.

Scoring is next, and the miniature was designed to make it teachable: ten customers cut into five bins means exactly two per bin, fully checkable by hand — and frequency was designed to break.

Code 6.3. Diagnose and resolve tied frequency scores

```
cf["R"] = pd.qcut(cf["recency_days"], 5, labels=[5,4,3,2,1]).astype(int)
cf["M"] = pd.qcut(cf["monetary"],     5, labels=[1,2,3,4,5]).astype(int)

# Frequency: predict the result before running this block.
try:
    cf["F"] = pd.qcut(cf["frequency"], 5, labels=[1,2,3,4,5]).astype(int)
except ValueError as e:
    print("qcut refused to cut frequency:", e)

# The declared repair (Section 6.5): three frequency bands cut at values
# the distribution actually takes, mapped onto the grid's 1-5 axis.
# Customers with the same order count always receive the same score.
freq_bands = pd.IntervalIndex.from_tuples(
    [(0, 1), (1, 3), (3, np.inf)], closed="right")
band_score = {freq_bands[0]: 1, freq_bands[1]: 3, freq_bands[2]: 5}
cf["F"] = (pd.cut(cf["frequency"], bins=freq_bands)
             .map(band_score).astype(int))

assert cf[["R", "F", "M"]].notna().all().all()
assert set(cf["F"]).issubset({1, 3, 5})
assert (cf.groupby("frequency")["F"].nunique() == 1).all()

print(cf[["recency_days", "frequency", "monetary", "R", "F", "M"]]
      .sort_values(["R", "F"], ascending=False))
```

Expected output: a ValueError message reading "Bin edges must be unique," then a ten-row table in which C005 scores R5 F3 M3, C009 scores R5 F1 M1, C001 scores R4 F5 M4, and C006 scores R1 F1 M4.

Input: the miniature feature table. Transformation: recency and monetary value are cut into quintiles, frequency is banded, and the scores are attached. Output: three ordinal columns and one instructive failure. The ten frequency values are 4, 3, 3, 2, 2, 2, 1, 1, 1, 1: four distinct values cannot supply five distinct bin edges, so qcut raises rather than inventing a cut. That error is the data telling you the quantity cannot support five honest bins, and the repair is a scoring-rule decision rather than a coding workaround.

The third assertion distinguishes this repair from the row-order tiebreaker an assistant would have reached for: it states that any two customers with the same frequency receive the same score, which is false under rank(method="first"). The bands are a declared rule and belong in the deliverable's written definitions — one order scores F1, two or three score F3, four or more score F5.

With scores in hand, the last step of Part A is the assignment that the opening case actually requires: every customer in exactly one named region, by a rule two analysts would apply identically.

Code 6.4. Assign every customer to a named grid region

```
# Table 6.3 as code: a complete lookup over all twenty-five cells.
RFM_GRID = {
    (5, 1): "New Customers",       (5, 2): "Potential Loyalists",
    (5, 3): "Loyal Customers",     (5, 4): "Champions",
    (5, 5): "Champions",
    (4, 1): "Potential Loyalists", (4, 2): "Potential Loyalists",
    (4, 3): "Loyal Customers",     (4, 4): "Champions",
```

```
    (4, 5): "Champions",
    (3, 1): "Needs Attention",      (3, 2): "Needs Attention",
    (3, 3): "Loyal Customers",      (3, 4): "Loyal Customers",
    (3, 5): "Loyal Customers",
    (2, 1): "Hibernating",          (2, 2): "Needs Attention",
    (2, 3): "At Risk",              (2, 4): "At Risk",
    (2, 5): "At Risk",
    (1, 1): "Hibernating",          (1, 2): "Hibernating",
    (1, 3): "At Risk",              (1, 4): "At Risk",
    (1, 5): "At Risk",
}

# Exhaustive and mutually exclusive, checked rather than asserted in prose.
assert len(RFM_GRID) == 25
assert set(RFM_GRID) == {(r, f) for r in range(1, 6)
                                for f in range(1, 6)}

cf["rfm_region"] = [RFM_GRID[(r, f)] for r, f in zip(cf["R"], cf["F"])]

assert cf["rfm_region"].notna().all()
assert len(cf["rfm_region"]) == len(cf)

print(cf[["R", "F", "M", "rfm_region"]]
      .sort_values(["R", "F"], ascending=False))
print(cf["rfm_region"].value_counts())
```

Expected output: C001 in Champions; C005, C008, and C003 in Loyal Customers; C009 in New Customers; C004 in Needs Attention; C002 and C010 At Risk; C006 and C007 Hibernating — three Loyal Customers, two At Risk, two Hibernating, and one each in Champions, Needs Attention, and New Customers.

Input: the scored miniature. Transformation: a dictionary lookup on the pair of scores. Output: one region per customer, and two assertions that make the grid's structural promise executable — twenty-five cells defined, and every customer assigned. A lookup written this way cannot develop the gaps and overlaps a prose rule set develops, because the second assertion fails the moment a cell goes missing.

Read the result against the raw extract before moving on. C002, the high-basket occasion shopper, carries the largest monetary total in the table and lands in At Risk, because the regions are drawn on recency and frequency alone — Section 6.5's closing argument made visible on ten rows, and exactly the limitation that sends the chapter to clustering.

6.13.2 Lab 6.1, Part B: The Build at Full Scale

Part B repeats the build on the certified transactions file and the customers table, per Table 6.5's full derivation rules. The window and the analysis date are declared in the first cell, the population rule is declared with them, the ratio and intensity features join the core three, and the customer attributes join as profiling holdouts. An AI assistant may draft this build using the specification prompt in Section 6.12's AI in Practice box; the reconciliation, not the drafting, is the graded discipline. The cells assume transactions_clean and customers are loaded and that order_date has been parsed to a true date.

Code 6.5. Build the full customer-feature table

```
analysis_date = pd.Timestamp("2026-06-30")
window_start  = analysis_date - pd.DateOffset(months=24)

# Certified figures for the DECLARED WINDOW, copied from the Chapter 4
# verification log before anything is computed.
CERT_REVENUE    = ...   # total revenue in the window
CERT_ORDERS     = ...   # distinct orders in the window
CERT_PURCHASERS = ...   # distinct customers with an order in the window
```

```
window = transactions_clean[
    (transactions_clean["order_date"] > window_start)
    & (transactions_clean["order_date"] <= analysis_date)]

# Key integrity FIRST. groupby drops rows whose key is missing and
# nunique() ignores missing values, so a missing customer_id could pass
# a "<= 1 customer per order" check by counting zero.
required = ["order_id", "customer_id", "order_date",
            "line_revenue", "channel"]
assert window[required].notna().all().all(),     "required transaction fields
contain missing values"
assert customers["customer_id"].is_unique
assert window["customer_id"].isin(customers["customer_id"]).all()

# Each order must carry exactly one customer, one date, one channel --
# otherwise the "first" and "min" below would silently pick a value.
for col, msg in [("customer_id", "an order has zero or multiple customers"),
                 ("order_date", "an order spans multiple dates"),
                 ("channel", "an order spans multiple channels")]:
    assert (window.groupby("order_id")[col]
            .nunique(dropna=False).eq(1).all()), msg

orders = (window
          .groupby(["order_id", "customer_id"], as_index=False)
          .agg(order_date=("order_date", "min"),
               order_revenue=("line_revenue", "sum"),
               channel=("channel", "first")))

assert len(orders) == window["order_id"].nunique()
assert np.isclose(orders["order_revenue"].sum(),
                  window["line_revenue"].sum())

# Discounted revenue stays at line grain: an order can mix discounted
# and full-price lines, so this cannot be computed from order totals.
disc = (window
        .assign(discounted=lambda d: d["line_revenue"]
                .where(d["discount_pct"] > 0, 0.0))
        .groupby("customer_id")
        .agg(discounted_revenue=("discounted", "sum")))

# POPULATION RULE (Section 6.6): one row per customer with at least one
# order in the window. Everyone else is counted, named, and routed --
# not clustered, and not silently dropped.
cf_full = (orders
           .groupby("customer_id")
           .agg(last_order=("order_date", "max"),
                frequency=("order_id", "nunique"),
                monetary=("order_revenue", "sum"),
                store_orders=("channel", lambda s: (s == "Store").sum()))
           .join(disc)
           .join(customers.set_index("customer_id")[["signup_date"]]))

# The complement of the clustered population is NOT "never purchased".
window_purchasers  = set(window["customer_id"])
observed_purchasers = set(transactions_clean["customer_id"].dropna())

never_observed = customers[
    ~customers["customer_id"].isin(observed_purchasers)]
inactive_outside_window = customers[
    customers["customer_id"].isin(observed_purchasers)
    & ~customers["customer_id"].isin(window_purchasers)]
```

```
# The three populations partition the customers table exactly.
assert (len(cf_full) + len(never_observed)
        + len(inactive_outside_window) == len(customers))

print("purchasing customers (clustered):", len(cf_full))
print("customers with no purchase in the analysis window:",
      len(customers) - len(cf_full))
print("  no purchase in observed history -> activation program:",
      len(never_observed))
print("  purchased only before the window -> lapsed, win-back:",
      len(inactive_outside_window))
```

Expected output: a clustered row count equal to the certified distinct-purchaser count; three population counts that add back to the customers table; and every assertion passing.

Input: the certified line-grain file and the customers dimension. Transformation: the declared window is applied, keys are validated, lines are collected to orders, discounted revenue is summed at line grain, and orders are collected to customers. Output: the level features, the raw material for the ratios, and a three-way population count. The window filter is stated as an interval, open at the start and closed at the end, so that boundary orders belong to exactly one side; and the discount roll-up runs on lines rather than orders, because an order that mixes a discounted dress with a full-price scarf is one order and two different facts.

The key checks come before the grouping for a reason worth knowing. pandas drops rows whose grouping key is missing, and nunique() ignores missing values by default, so an order whose customer_id is missing would disappear from the grouped frame while a check written as nunique().le(1) counted zero and passed. Running notna() first, then asserting eq(1) with dropna=False, closes both doors. The date and channel checks close a third: the aggregation below takes the minimum date and the first channel, which would quietly manufacture a consistent order out of an inconsistent one.

The population arithmetic is the correction the opening case needs. Subtracting the feature table's row count from the customers table counts everyone with no purchase in this window, which is not the same population as customers who have never purchased: it also includes customers whose only orders predate the window. The two go to different places -- the first to the always-on activation program, the second to win-back inside the fall program -- so the cell separates them and asserts that the three counts partition the base. On a file holding only twenty-four months, even never_observed means no purchase in observed history rather than none ever, and the deliverable says so.

Code 6.6. Derive the ratio and intensity features, and reconcile

```
# Exposure is observed time inside the window, not calendar tenure, and
# it carries a declared floor: a customer observed for nine days cannot
# support a stable monthly rate.
cf_full["recency_days"] = (analysis_date
                           - cf_full["last_order"]).dt.days
observed_days = (analysis_date
                 - cf_full["signup_date"].clip(lower=window_start)).dt.days
cf_full["exposure_months"] = np.maximum(observed_days / 30.44, 1.0)

cf_full["orders_per_month"]  = (cf_full["frequency"]
                                / cf_full["exposure_months"])
cf_full["revenue_per_month"] = (cf_full["monetary"]
                                / cf_full["exposure_months"])
cf_full["aov"]            = cf_full["monetary"] / cf_full["frequency"]
cf_full["discount_share"] = (cf_full["discounted_revenue"]
                             / cf_full["monetary"])
cf_full["store_share"]    = (cf_full["store_orders"]
                             / cf_full["frequency"])
cf_full = cf_full.drop(columns=["last_order", "signup_date",
                                "store_orders", "discounted_revenue"])
```

```
# Profiling holdouts join last, and are never clustered on.
attrs = customers.assign(
    tenure_days=lambda d: (analysis_date - d["signup_date"]).dt.days)
holdouts = ["tenure_days", "loyalty_tier", "app_user", "email_opt_in",
            "home_metro", "acquisition_channel", "age_band"]
cf_full = cf_full.join(attrs.set_index("customer_id")[holdouts])

# The three certified equalities of Section 6.6, plus the edge cases.
assert len(cf_full) == CERT_PURCHASERS
assert cf_full["frequency"].sum() == CERT_ORDERS
assert np.isclose(cf_full["monetary"].sum(), CERT_REVENUE, atol=0.01)
assert (cf_full["recency_days"] >= 0).all()
assert cf_full[["aov", "discount_share", "store_share",
                "orders_per_month",
                "revenue_per_month"]].notna().all().all()
# Shares summed from rounded line revenue need a tolerance, not a bare 1.
tol = 1e-9
assert cf_full["discount_share"].between(-tol, 1 + tol).all()
assert cf_full["store_share"].between(-tol, 1 + tol).all()
assert (cf_full["exposure_months"] >= 1.0).all()
assert np.allclose(cf_full.loc[cf_full["frequency"] == 1, "aov"],
                   cf_full.loc[cf_full["frequency"] == 1, "monetary"])

print(cf_full[["recency_days", "frequency", "monetary", "aov",
               "orders_per_month", "revenue_per_month",
               "discount_share", "store_share"]].describe().round(2))
```

Expected output: an eight-column summary with no missing values, minimum recency of 0 or more, shares bounded by 0 and 1, and nine assertions passing.

Input: the aggregated customer table. Transformation: recency from the declared analysis date, exposure from the later of signup and the window start, the two intensity rates, and the three shape ratios. Output: the completed clustering inputs plus the holdouts. Note what exposure_months does and does not do. It uses signup date — the very column Table 6.5 holds out — as a denominator, deliberately: dividing by exposure reduces tenure's mechanical effect on lifetime totals and puts it on screen beside them, while clustering on tenure is how that same effect gets rediscovered and mistaken for a finding (Section 6.6). It does not remove the effect, because frequency and monetary remain in the specification and remain bounded by exposure. The floor is applied to the months, not to the days, so the minimum exposure is exactly 1.0 month and the assertion says so — an earlier draft floored the day count at 30 and produced a minimum of 0.986, which is the kind of quiet discrepancy between a stated rule and its implementation this chapter exists to catch.

The three certified equalities are the chapter's known-totals check in feature-table form, and the row-count equality is also the population statement: the feature table is smaller than the customers table by exactly the no-purchase-in-window count printed by Code 6.5, and that difference is reported, split, and named rather than absorbed.

VERIFICATION CHECK

Before you run: copy the three certified figures for the declared window from your Chapter 4 verification log into the notebook as constants, and predict in writing that the feature table's row count equals the certified purchaser count and is smaller than the customers table's row count — then state why in one sentence. Predict two more structural facts: recency_days has no negative values, and every customer with frequency = 1 has aov equal to monetary exactly.

After you run: all nine assertions pass, Code 6.5's three population counts add back to the customers table's row count, the minimum exposure_months is exactly 1.0, and the summary shows no missing values in any clustering input.

> Investigate if: your finished table disagrees with the shipped customer_features answer key on any shared column. A merged comparison should agree everywhere; a disagreement is a derivation-rule difference to locate and name, not to average away. Check the window boundary first — it is the most common source.

Part B closes with the full-scale grid, which reuses Code 6.4's lookup unchanged. That reuse is the point: the assignment rule is the same object at ten customers and at eight thousand.

Code 6.7. Score and read the full-scale RFM grid

```
# Recency and monetary value support quintiles at scale; frequency
# still does not, so the banded rule of Code 6.3 carries over unchanged.
cf_full["R"] = pd.qcut(cf_full["recency_days"].rank(method="average"),
                       5, labels=[5, 4, 3, 2, 1]).astype(int)
cf_full["M"] = pd.qcut(cf_full["monetary"].rank(method="average"),
                       5, labels=[1, 2, 3, 4, 5]).astype(int)
cf_full["F"] = (pd.cut(cf_full["frequency"], bins=freq_bands)
                  .map(band_score).astype(int))

cf_full["rfm_region"] = [RFM_GRID[(r, f)]
                         for r, f in zip(cf_full["R"], cf_full["F"])]

# Every customer assigned, exactly once.
assert cf_full["rfm_region"].notna().all()
assert len(cf_full["rfm_region"]) == len(cf_full)

grid = (cf_full.groupby("rfm_region")
        .agg(customers=("monetary", "size"),
             revenue=("monetary", "sum"))
        .assign(customer_share=lambda d: d["customers"]
                / d["customers"].sum(),
                revenue_share=lambda d: d["revenue"]
                / d["revenue"].sum()))

assert np.isclose(grid["revenue"].sum(), CERT_REVENUE, atol=0.01)
assert grid["customers"].sum() == len(cf_full)

print(grid.round(3))
```

Expected output: one row per occupied region, customer and revenue shares each summing to 1.000, and the region revenues reconciling to the certified window total.

Input: the full feature table. Transformation: the declared scoring rule and the twenty-five-cell lookup. Output: the named grid with sizes and revenue shares. The two assertions restate the exhaustiveness promise at scale and add the reconciliation that makes the revenue shares readable: a grid whose revenue does not add back to the certified window total has lost customers somewhere between the feature table and the summary.

Close Part B by writing three insight statements per Section 5.9 from the grid — at least one of which must name something the grid cannot see, rehearsing Section 6.5's closing argument and setting up Lab 6.2.

6.13.3 Lab 6.2: Discovery, Validation, and the Answer Key

Lab 6.2 clusters. The inputs are the eight behavioral features of Table 6.5; the holdouts stay held out; and the lab's structure is the chapter's argument in code: scale, sweep, choose, profile, validate, test the specification, compare, and deploy.

Code 6.8. Standardize the clustering features

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, adjusted_rand_score

features = ["recency_days", "frequency", "monetary",
            "orders_per_month", "revenue_per_month",
            "aov", "discount_share", "store_share"]

# The scaler is fitted once and KEPT. Section 6.10 needs this object.
scaler = StandardScaler()
X = scaler.fit_transform(cf_full[features])

assert np.allclose(X.mean(axis=0), 0, atol=1e-9)
assert np.allclose(X.std(axis=0), 1, atol=1e-9)

print(np.round(X.mean(axis=0), 6))   # eight zeros
print(np.round(X.std(axis=0), 6))    # eight ones
print(pd.DataFrame(X, columns=features).corr().round(2))
```

Expected output: eight zeros, eight ones, and a correlation matrix in which frequency and monetary are strongly related to each other and to their per-month counterparts.

Input: the eight clustering features. Transformation: z-scores. Output: the scaled matrix, plus the two facts that define a z-score, printed rather than assumed. The correlation matrix is not decoration. It is the visible form of Section 6.7's last argument: standardization has equalized the eight features' scales and done nothing whatever about the fact that several of them carry the same information. Code 6.13 measures what that costs.

A useful side computation, echoing Section 6.7: pick one high-monetary and one high-frequency customer and confirm that scaling changed which of their differences dominates the distance between them.

Code 6.9. Evaluate candidate values of k

```
sweep = []
for k in range(2, 9):
    km = KMeans(n_clusters=k, n_init=10, random_state=42).fit(X)
    sweep.append((k, km.inertia_, silhouette_score(X, km.labels_)))

sweep = pd.DataFrame(sweep,
                     columns=["k", "inertia", "mean_silhouette"])

# Inertia should be non-increasing in a well-optimized sweep. A rise is a
# WARNING to inspect, not proof of a bug: each fit is a separate
# optimization and can settle into a worse local minimum.
if not sweep["inertia"].is_monotonic_decreasing:
    print("WARNING: inertia rose; refit with more starts before "
          "reading the elbow.")

print(sweep.round(3))
```

Expected output: seven rows; inertia falling steeply from k = 2 and then flattening; mean silhouette values that do not necessarily peak where the elbow bends.

Input: the scaled matrix. Transformation: seven independent fits. Output: the two instruments of Section 6.9, side by side. Note the diagnostic rather than the assertion. An earlier draft of this lab asserted that inertia must strictly decrease; that is too strong on both counts. At the optimum, inertia is non-increasing rather than strictly decreasing — an added centroid that buys no improvement leaves it flat — and in practice a higher-k fit can land in a worse local minimum than a lower-k fit, which produces a rise that means “inspect and refit,” not “the code is broken.”

Then apply the braid. Shortlist the elbow's neighborhood and the silhouette's nominees, and carry the shortlist — not a single winner — into profiling. The choice of k = 4 for the cells that follow must be argued in a short markdown cell citing all three legs of Table 6.6, including the four-journey actionability constraint. “The answer key has four” is not one of the legs, and the cell should read as if no answer key existed.

VERIFICATION CHECK

Before you run: predict that inertia will be non-increasing across the sweep and say why in one sentence. Then predict which k the silhouette will crown, knowing Section 6.9's warning and Chapter 5's evidence of one dominant urban-suburban divide: the defensible prediction is that a small k scores at or near the top even though the designed structure is finer. Record the prediction before running, because the point of the exercise is not to be right but to notice when the instrument disagrees with the strategy.

After you run: the inertia column falls and flattens; the silhouette column has a maximum you can name; and the two instruments nominate either the same k or a shortlist of two or three.

Investigate if: the silhouette's winner is k = 2 and you were about to deploy it. Two segments cannot carry the discount-dependence and lifecycle distinctions the treatments exist to exploit, so the silhouette's verdict is evidence about geometry that judgment is entitled to overrule — in writing, with the reason stated.

Code 6.10. Fit and profile the selected solution

```
K = 4
model = KMeans(n_clusters=K, n_init=10, random_state=42)
labels = model.fit_predict(X)
cf_full["cluster"] = labels

assert len(np.unique(labels)) == K

centroids_std = pd.DataFrame(model.cluster_centers_, columns=features)
centroids_std.index.name = "cluster"

# A centroid IS the mean of its cluster, up to the optimizer's
# convergence tolerance. This is the audit of Section 6.12, executed.
assert np.allclose(centroids_std.values,
                   scaler.transform(
                       cf_full.groupby("cluster")[features].mean()),
                   atol=0.01)

native = (cf_full.groupby("cluster")[features]
          .agg(["median", "mean"]).round(2))
sizes = cf_full["cluster"].value_counts().sort_index()

assert sizes.sum() == len(cf_full)

print(sizes)                      # sizes first, per Section 5.6
print(centroids_std.round(2))     # what defines each cluster
print(native)                     # what to say in the meeting
```

Expected output: four cluster sizes summing to the feature table’s row count; a centroid table in standard-deviation units; and a native table carrying medians beside means for every feature.

Input: the scaled matrix. Transformation: one fit at the chosen k. Output: the profile in both vocabularies. Read the standardized table first to see what defines each cluster — a value of −0.8 on recency_days means the cluster's

members bought more recently than average by four fifths of a standard deviation — then read the native table for the deck, because standardized units are for analysts and days and dollars are for decisions.

Medians travel beside means for the reason Section 5.4 established: a cluster's monetary centroid can be propped by a handful of whales, and a mean its own median contradicts is a warning that the cluster is not one story. Before touching the holdouts, write the predictions — for each cluster, one sentence on how it should differ on loyalty_tier, app_user, home_metro, acquisition_channel, and tenure_days if its behavioral profile means what it appears to mean.

Code 6.11. Test seeded and subsample stability

```
# Each fit below is the best of ten internal initializations, so these
# are five INDEPENDENTLY SEEDED FITS, not five random starting points.
seed_ari = {}
for seed in [7, 21, 99, 2026, 31337]:
    alt = KMeans(n_clusters=K, n_init=10,
                 random_state=seed).fit_predict(X)
    seed_ari[seed] = adjusted_rand_score(labels, alt)

# Subsample refits: 80 percent of customers, compared on the customers
# both solutions saw. The index is relabeling-invariant, so no alignment
# step is needed and no dominant-cell counting can mislead.
sub_ari = {}
rng = np.random.default_rng(42)
for i in range(2):
    idx = rng.choice(len(X), size=int(0.8 * len(X)), replace=False)
    sub = KMeans(n_clusters=K, n_init=10,
                 random_state=42).fit_predict(X[idx])
    sub_ari[f"subsample_{i + 1}"] = adjusted_rand_score(labels[idx], sub)

print("seeded ARI:  ", {s: round(v, 3) for s, v in seed_ari.items()})
print("subsample ARI:", {s: round(v, 3) for s, v in sub_ari.items()})
print("minimum observed:",
      round(min(list(seed_ari.values()) + list(sub_ari.values())), 3))
```

Expected output: seven indices no greater than 1.0, with a stable solution close to 1.0 throughout; values near 0 would indicate chance-level agreement and negative values worse-than-chance agreement.

Input: the scaled matrix and the baseline labels. Transformation: seven refits under perturbation. Output: seven agreement scores. The adjusted Rand index asks, over all pairs of customers, how often two solutions agree about whether the pair belongs together, corrected for the agreement chance would produce (Hubert & Arabie, 1985). It is 1.0 for identical partitions and near 0.0 for unrelated ones, and it can go negative — to a lower bound of −0.5 — when two partitions agree less than chance would predict, which is why the expected output states a ceiling rather than a range (Pedregosa et al., 2011).

The index replaces the procedure most practitioner treatments use — cross-tabulate the two runs and count the customers outside each row's largest cell — and the replacement is not cosmetic: that procedure assumes the runs correspond one to one, and two baseline clusters can both have their largest overlap with the same cluster in the refit, reporting a stability that does not exist. Where a customer-level reassignment rate is wanted for the deck, align the labels first with an explicit one-to-one assignment, then count, and report the index alongside it. In the deck's plain language: “we refit from five independent seeds and on two random 80 percent subsets; the partitions agreed at an adjusted Rand index of at least [value].” A solution that holds its shape has earned Section 6.10's adjective; one that churns has not, whatever its silhouette said.

Code 6.12. Profile the withheld attributes

```
hold_num = (cf_full.groupby("cluster")
            .agg(customers=("tenure_days", "size"),
                 tenure_days_median=("tenure_days", "median"),
```

```
                  app_user=("app_user", "mean"),
                  email_opt_in=("email_opt_in", "mean")))

assert hold_num["customers"].sum() == len(cf_full)
print(hold_num.round(3))

# crosstab drops missing values silently and the rows still sum to 1.0,
# so the assertion would pass while hiding a real difference. age_band is
# partly missing by design (Table 6.1), and missingness can itself differ
# across clusters -- so it is profiled as a category, not discarded.
for col in ["home_metro", "loyalty_tier",
            "acquisition_channel", "age_band"]:
    profiled = cf_full[col].astype("string").fillna("Missing")
    tab = pd.crosstab(cf_full["cluster"], profiled, normalize="index")
    assert np.allclose(tab.sum(axis=1), 1.0)
    print(col)
    print(tab.round(3))
```

Expected output: a numeric holdout table with one row per cluster, then four row-normalized cross-tabulations whose rows each sum to 1.000, with an explicit Missing column wherever the attribute is incomplete.

Input: the clustered feature table. Transformation: group by cluster, summarize the columns the algorithm never saw. Output: the holdout profile. Row-normalize rather than cell-normalize, per Section 5.5's direction rule: the comparison being made is across clusters, so the percentages run within each cluster. Filling missing values with an explicit Missing category matters more here than anywhere else in the chapter. Dropping them leaves every row summing to 1.000 and the assertion passing while a cluster that is half unknown on age band looks exactly like one that is fully observed — and in a holdout check, systematically different missingness is itself a finding.

Now grade the predictions you wrote before Code 6.11 ran, and grade them honestly in both directions. Confirmed predictions are one layer of external-variable evidence, and they are worth a sentence in the deck. They are not proof that the clusters are natural kinds: the holdout attributes are correlated with the clustering inputs, so agreement is convergent rather than independent (Section 6.10). And nothing here speaks to differential response, which is a causal claim no clustering can support. Failed predictions are findings: a cluster you expected to be loyalty-enrolled and that turns out not to be has just told you the behavioral features were incomplete.

Code 6.13. Test feature-set sensitivity

```
# Standardization equalizes scale; it does not remove redundancy.
# aov is monetary / frequency, and each per-month rate is a level
# feature / exposure. This cell measures what those dependencies cost.
SPECS = {
    "base (8 features)": features,
    "without aov": [f for f in features if f != "aov"],
    "without level features":
        [f for f in features if f not in ("frequency", "monetary")],
    "without intensity features":
        [f for f in features if not f.endswith("_per_month")],
}

sens = {}
for name, cols in SPECS.items():
    Xs = StandardScaler().fit_transform(cf_full[cols])
    lab = KMeans(n_clusters=K, n_init=10,
                 random_state=42).fit_predict(Xs)
    sens[name] = adjusted_rand_score(labels, lab)

assert np.isclose(sens["base (8 features)"], 1.0)
```

```
for name, v in sens.items():
    print(f"{name:<28} ARI vs base: {v:.3f}")
```

Expected output: 1.000 for the base specification against itself, and three values at or below 1.000 for the alternatives; an alternative may also reach 1.000 if removing those features leaves every assignment unchanged.
Input: the feature table. Transformation: three refits on reduced feature sets, each compared with the base solution by adjusted Rand index. Output: a sensitivity report. The first line is a control: a specification compared with itself must return exactly 1.0, and if it does not, the comparison is broken before any conclusion is drawn.

The purpose is not to select a winner automatically but to make visible that feature selection is part of the definition of distance, and therefore of "alike." If dropping aov moves a large share of customers, aov was carrying real weight and its inclusion is a modeling decision to be defended rather than defaulted; if it moves almost nobody, the composite was redundant and the simpler specification is easier to govern. Either way the honest deliverable reports the comparison, not the winner.

Code 6.14. Compare with the designed answer key

```
designed = (customers.set_index("customer_id")["segment_designed"]
            .reindex(cf_full.index))
assert designed.notna().all(), "answer key misses some customers"

key = pd.crosstab(designed, cf_full["cluster"])
assert key.values.sum() == len(cf_full)

print(key)
print("ARI against the designed segments:",
      round(adjusted_rand_score(designed, labels), 3))
```

Expected output: a four-by-four cross-tabulation whose cells sum to the feature table's row count, and a single agreement index.
Input: the clustered table and the shipped answer key. Transformation: one cross-tabulation and one index. Output: the comparison only a designed dataset allows. Read the cross-tabulation the way Section 6.10 taught: because cluster numbering is arbitrary, the signature of strong recovery is a permutation pattern — each designed segment concentrated in one cluster column — rather than a diagonal, and the index summarizes the pattern without requiring the labels to line up.

The off-diagonal customers are the finding, not the failure. The specification planted four segments — Urban Loyal Core, Suburban Occasion, Discount Trend Follower, and New/At-Risk — separable on behavioral structure but with deliberate overlap, so recovery requires judgment rather than falling out automatically. Profile a handful of the misassigned customers individually: that is what "the boundaries between segments are judgment calls" looks like one customer at a time.

Code 6.15. Build the segment-attractiveness matrix

```
NAMES = {0: "...", 1: "...", 2: "...", 3: "..."}   # after profiling
cf_full["segment"] = cf_full["cluster"].map(NAMES)
assert cf_full["segment"].notna().all()

# Table 6.8's row is DISCOUNT SHARE OF REVENUE -- a pooled ratio. The
# mean of customer-level shares would weight every customer equally and
# answer a different question, so discounted revenue is rebuilt and the
# ratio is taken on the totals (the averaged-average rule of Section 5.3).
cf_full["discounted_revenue"] = (cf_full["monetary"]
                                 * cf_full["discount_share"])

attract = (cf_full.groupby("segment")
           .agg(customers=("monetary", "size"),
                revenue=("monetary", "sum"),
```

```
                discounted_revenue=("discounted_revenue", "sum"),
                revenue_per_customer=("monetary", "mean"),
                revenue_per_exposure_month=("revenue_per_month", "mean"),
                median_tenure_days=("tenure_days", "median"),
                app_reachable=("app_user", "mean"),
                email_reachable=("email_opt_in", "mean")))
attract["discount_share_of_revenue"] = (attract["discounted_revenue"]
                                        / attract["revenue"])
attract["customer_share"] = (attract["customers"]
                             / attract["customers"].sum())
attract["revenue_share"] = attract["revenue"] / attract["revenue"].sum()

# The matrix is a partition of the base and of the window's revenue.
assert attract["customers"].sum() == len(cf_full)
assert np.isclose(attract["revenue"].sum(), CERT_REVENUE, atol=0.01)
assert np.isclose(attract["customer_share"].sum(), 1.0)

print(attract.round(3))
```

Expected output: one row per named segment, customer and revenue shares each summing to 1.000, revenue reconciling to the certified window total, and a discount share of revenue that differs from the average of the segment's customer-level shares.

Input: the named, clustered table. Transformation: one grouped summary. Output: the profile-arithmetic rows of Table 6.8. Two figures deserve attention. revenue_per_exposure_month is the size-adjusted companion to revenue_per_customer, and comparing the two is how a lifecycle segment stops being mistaken for a weak one: a segment can trail badly on lifetime revenue and lead on revenue per month of exposure, which is a different sentence with a different budget attached. And discount_share_of_revenue is computed as a pooled ratio rather than as the mean of customer-level shares, because Table 6.8 names it a share of revenue: averaging the customer shares gives a thousand-dollar customer and a forty-dollar customer equal votes, which is Section 5.3's averaged average wearing a margin costume. It stands in for margin quality until finance supplies real unit costs, and it is labeled a proxy in the deck for exactly that reason.

Two rows of Table 6.8 are deliberately absent from this cell. The same-age trajectory is evidence rather than judgment, but it needs the period axis of a cohort table and must be merged in from the Chapter 5 analysis; leaving it blank here and filling it there is the honest arrangement, and forgetting it is how a lifecycle segment gets judged on lifetime totals after all. Cost to serve, strategic fit, the evidence-versus-judgment declaration, and the recommended priority are not computable at all: write them by hand, mark them as judgment, and defend them in the meeting, because a matrix that silently mixes measured cells with asserted ones is the failure that declaration exists to prevent.

Code 6.16. Save and apply the operating assignment model

```
# Deployment needs the fitted objects, not thresholds read off a table.
SEGMENTATION = {
    "version": "2026-fall-v1",
    "analysis_date": analysis_date,
    "window_start": window_start,
    "population": "customers with >= 1 order in the window",
    "features": features,
    "scaler": scaler,          # the fitted means and standard deviations
    "model": model,            # the fitted centroids
    "names": NAMES,
}

def assign_segments(new_features, spec=SEGMENTATION):
    """Assign with the FITTED scaler and model. Never refit here:
    refitting would silently redraw the segments."""
    missing = [f for f in spec["features"]
               if f not in new_features.columns]
```

```
    assert not missing, f"missing features: {missing}"
    Xn = spec["scaler"].transform(new_features[spec["features"]])
    return pd.Series([spec["names"][c]
                      for c in spec["model"].predict(Xn)],
                     index=new_features.index, name="segment")

# Reproducing the fitted assignments is the deployment check.
assert (assign_segments(cf_full) == cf_full["segment"]).all()
sample = cf_full.sample(500, random_state=1)
assert (assign_segments(sample) == sample["segment"]).all()

# An ILLUSTRATIVE CANDIDATE rule for the campaign platform, written
# after profiling. Branches return SEGMENT NAMES, never NAMES[i]:
# cluster numbers are arbitrary, so a discount branch has no reason to
# land on cluster 0. Whether this rule is deployable is decided by the
# agreement it measures, not by the fact that it was written.
rule = np.where(cf_full["discount_share"] > 0.50, "Discount-Dependent",
        np.where(cf_full["store_share"] > 0.60, "Suburban Occasion",
         np.where(cf_full["orders_per_month"] > 0.50, "Urban Loyal Core",
                  "New / At-Risk")))
assert set(np.unique(rule)) <= set(NAMES.values())

agreement = (rule == cf_full["segment"]).mean()
print(f"candidate rule agrees with the model on {agreement:.1%} "
      "of customers")
print(pd.crosstab(cf_full["segment"], rule))
```

Expected output: both deployment assertions passing, one agreement percentage, and a cross-tabulation locating every customer on whom the rule and the model disagree.

Input: the fitted scaler, the fitted model, and the feature table. Transformation: a saved specification and an assignment function that transforms with the saved scaler and predicts with the saved centroids. Output: reproducible assignments and an honest account of the approximation the campaign platform will run.

The two deployment assertions are the whole point of the first half of the cell. The first says the saved objects reproduce the fitted assignments exactly; the second says they do so for an arbitrary subset, which is what assigning next month's new customers amounts to. Neither would pass if the function refitted — and refitting is exactly what a well-meaning engineer will do when the notebook is handed over without the objects.

The threshold rule in the second half is a candidate, not a conclusion, and two things about it are easy to get wrong. Its branches return segment names rather than NAMES[0] and NAMES[1], because cluster numbers carry no meaning: the cluster that turns out to be discount-dependent may be numbered 2, and a rule that maps a discount branch onto cluster 0 by construction is measuring its own indexing. And the agreement figure is the test, not the footnote. SQL is what the campaign platform speaks, so a simplified rule is often necessary; but if the measured agreement comes back low, the correct conclusion is that this rule is not an acceptable surrogate for the model, and the response is to find better thresholds, add a branch, or push the platform to consume scored assignments rather than recompute them. Publish the number beside the rule and keep the cross-tabulation: the customers in its off-diagonal cells are the ones who would receive the wrong journey.

The lab's final deliverable is the CRM manager's one-page treatment map per Section 6.11: four named segments with their written assignment rules and the saved model that those rules approximate, a journey and a priority for each, the customers with no purchase in the window split between activation and win-back per Code 6.5, the evidentiary basis in one line each, and the validation results — stability indices, feature-set sensitivity, and holdout agreement — as the map's footer. The sign-off meeting's first question is “why should we believe these four groups are real?”, and the footer is the answer. Deliverables for both labs: the notebook, the reconciliation log, the written scoring and population rules, the saved segmentation specification, the treatment map, and the AI-Use Appendix (Appendix D) documenting every assistant exchange, including at least one delegated step you corrected and why.

6.14 Marketing Interpretation and Managerial Insight

The lab's outputs are tables; the sign-off meeting runs on sentences. This section translates, and — per this guide's standing practice — it does so partly by exhibiting a wrong managerial reading and correcting it, because segmentation's characteristic misreadings are more expensive than description's: a misread summary misinforms a meeting, while a misread segmentation gets built into journeys, budgets, and triggers that run automatically for a year.

The wrong reading, and it will be offered in the meeting by someone reasonable: “Cluster 4 — the New/At-Risk group — has the lowest monetary value, the lowest frequency, and the weakest repeat behavior of the four segments. It is our worst-performing segment. The efficient move is to zero out its journey and reallocate the fourth slot's budget to Champions.” Every clause before the conclusion is true, and the conclusion is a trap with two jaws. The first jaw is arithmetic: the segment's lifetime totals are bounded by exposure. Many of its members are StyleCraft's most recent signups, and a customer nine weeks old cannot have accumulated two years of orders or two years of spend, so “lowest monetary and frequency” is partly what recency of arrival looks like at the customer grain — the tenure trap that Section 6.6 mitigated by adding the intensity features beside the lifetime totals rather than removing it, and that Chapter 5's cohort discipline exists to catch. The lab's own attractiveness matrix supplies the test: compare revenue per exposure-month beside revenue per customer, and compare the segment's early trajectory against older cohorts at the same age, per a Table 5.7-style cohort view. Lifetime totals do not answer the performance question; they restate the tenure question.

The second jaw is compositional. The newest customers concentrate, mechanically, wherever acquisition is newest — including the Wave 4 stores of the expansion. A treatment decision that abandons “low-value” new customers is, in substantial part, a decision to abandon the newest markets for being new, reached without ever mentioning them. And there is a third thing the wrong reading misses, which the segment's name has been carrying since Section 6.6: this cluster is a lifecycle position rather than a customer type, and the position it names is occupied by two strategically opposite populations. Genuinely new customers have unknown future value and a clear treatment — onboard them. Long-tenured customers who have gone quiet have known past value and a different treatment — win them back. The corrected reading holds all three points at once: “Cluster 4 is a lifecycle stage, not a quality verdict. Its observed value is low partly because its observed time is short; its members split into a new population and a lapsed one, which the journey must separate on tenure before it sends a single message — a branch inside the segment's one slot, not a fifth program; and its performance metric is same-age cohort progression and revenue per exposure-month, not lifetime spend.”

The corrected reading generalizes into the section's positive discipline: a segment profile supports sentences about what each group is and does, and it disciplines sentences about what to do. The treatment map earns managerial language when each row follows the insight-statement form of Section 5.9 — comparison, magnitude in decision units, connection to the named decision, verbs that stop at the evidence. “The Urban Loyal Core is roughly [share] of purchasing customers and [share] of revenue in the window, reaches near-fully through app and email, and is the base's retention engine — its journey should reward and protect, and discounting into it spends margin on behavior we already receive” survives scrutiny. “Targeting Champions will grow revenue 15 percent” does not — no analysis in this chapter measures what any treatment causes, and crossing that boundary requires the experimental evidence Part II reaches later. Between that evidence and this chapter stands a nearer question the meeting will also ask: what actually moves these customers — which levers, at what strength? Segments describe who; the drivers question is the next chapter's, and the treatment map should say so rather than improvise.

One more translation deserves its paragraph, because it converts this chapter's most technical work into its most persuasive slide: the validation footer. “We refit the analysis from five independent seeds and on two random 80 percent subsets of customers; the partitions agreed at an adjusted Rand index of at least [value], where 1.0 means identical partitions, 0 means chance agreement, and a negative value would mean worse than chance” is a sentence a non-technical executive can weigh, and so is “we built the groups from purchasing behavior only, then checked attributes the algorithm never saw.” State the footer's limits in the same breath, because the limits are what make the rest of it credible: stability establishes that the partition is reproducible, not that it is meaningful; holdout agreement establishes that the groups differ on attributes beyond their inputs, not that they are natural types or that they will

respond differently to treatment. The honest headline is therefore not “the customer base contains four kinds of customers” but “these four groups are reproducible, interpretable, and different enough on attributes we withheld to be worth treating differently — and the first year of treatment is how we find out whether they respond differently too.” That sentence is longer, and it is the one that survives the follow-up question.

6.15 Business Analytics in Practice

This section turns from the fictional case to how segmentation operates in industry — where it is among the oldest analytic disciplines in marketing and among the most routinely fumbled, usually at the joint between the statistics and the organization. The three vignettes below are composite: they synthesize recurring patterns in retail, financial services, and consumer marketing teams rather than reporting on any one named organization, and they are offered as situations a new analyst will recognize rather than as claims about how often they occur.

6.15.1 Nine Clusters, Three Treatments

The first vignette concerns a large retail bank. The bank's analytics group, well staffed and technically strong, delivered a nine-cluster segmentation of its retail customer base: statistically defensible, stable under refit, beautifully profiled, each cluster with a name, a persona sheet, and a deck of its own. The deployment meeting killed it in an hour, for a reason no silhouette score could see. The bank's CRM and campaign infrastructure could execute exactly three differentiated treatment programs — three creative streams, three offer structures, three measurement frames — and no budget existed to change that within the planning year. Nine segments mapped onto three treatments is, operationally, three segments with six ghosts.

The rework that followed inverted the process. The treatment capacity was written into the analytic specification as a design input, the clustering was rerun and evaluated at small k, and the deliverable became three coarse, deployable segments each with genuinely different economics — a solution the first project's authors would have called statistically timid and the campaign calendar called usable. The practice lesson is Sections 6.2 and 6.11 fused: actionability is not the last criterion applied to a finished segmentation; in mature organizations it is the first constraint written into the specification, and the number of treatments the organization can execute is as binding as any property of the data.

6.15.2 RFM as a Production System

The second vignette is RFM as a production system rather than an analysis. Tiered loyalty programs in fashion, beauty, and specialty retail run on machinery any reader of Section 6.5 would recognize: customers are scored on recency, frequency, and spend on an automated cadence; tier assignment and tier movement are rule-driven consequences of the scores; and the scores fire triggers — a recency threshold crossed without a purchase launches the win-back sequence, a frequency milestone unlocks the reward, a lapsed high-monetary customer routes to the retention offer with the richest allowable incentive.

Practitioners who operate these systems describe the analytic work as largely governance rather than modeling. The scoring definitions are metric definitions in exactly Section 3.5.1's sense — owned, documented, and versioned — because a silent change to the recency window reassigns thousands of customers overnight and quietly rewrites the meaning of every downstream report. Threshold changes are proposed, impact-sized against a trailing window, and change-logged in the manner Section 5.13.3 described for funnel definitions. And the recurring operational failures are measurement failures — a timezone bug in last-purchase dates, a returns policy that made monetary value negative for edge-case customers — rather than statistical ones.

The same governance applies to the model-based segmentations of Sections 6.7 through 6.10, and the discipline there is worth spelling out because it is where new analysts are most often surprised. A production segmentation is a versioned release: the feature definitions, the fitted scaler, the fitted centroids, and the population rule ship together under a version number, and new customers are scored against that release rather than against a fresh fit. The release is refreshed on a declared cadence rather than continuously, because a segmentation that silently redraws itself every night makes every trend chart that uses it uninterpretable. Between refreshes the assignments are monitored for drift: the share of customers landing in each segment is tracked, and a material move triggers investigation before it

triggers a refit, since the usual cause is an upstream data change rather than a change in customers. And a release is eventually retired — redrawn on new features, or replaced by a scheme keyed to a new strategy — with a documented mapping from old segments to new, because every dashboard, trigger, and target that referenced the old names has to be moved. The lesson reinforces this chapter's a priori road and extends Chapter 3's closing thread: an operating segmentation is a set of written definitions and fitted objects running on a schedule, and whoever governs them governs the program.

6.15.3 Personas and Statistical Segments

The third vignette concerns the reconciliation problem the opening case staged: personas versus statistical segments. Consumer organizations of any size frequently hold both — a brand or insights team maintains qualitative personas, vivid and motivating, while an analytics team maintains data-driven segments, measurable and operable — and the default relationship between the two artifacts is mutual disregard, with creative teams planning against characters no database can find and CRM teams targeting groups no storyteller can love. The strategy literature has documented the failure at the persona end: segmentations built for messaging drift toward demographic and lifestyle portraits that neither predict behavior nor guide the decisions the segmentation was commissioned to serve (Yankelovich & Meer, 2006).

The reconciliation practice that mature organizations converge on runs through exactly this chapter's machinery: treat the personas as hypotheses, profile the statistical segments on every attribute the persona narratives invoke, and test the mapping. Sometimes a persona turns out to be a recognizable region of a real segment and inherits its size, value, and reachability; sometimes a beloved persona maps to nobody, which is a finding the focus group could never have produced. Where the mapping holds, the persona survives as the communication layer — the name and face the creative team writes to — wrapped around a measured membership the CRM team can actually target; where it fails, one artifact yields. The lesson is Section 6.10's naming discipline at organizational scale: stories and statistics are reconciled by profiling, not by seniority.

6.15.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into a single expectation: the segmentation work you will actually be asked to defend is rarely the clustering. It is the specification that wrote treatment capacity in as a constraint, the definitions document that lets the scoring run unattended for a year, the versioned release that lets next month's customers be assigned without redrawing last month's, the profile that told the brand team which of their personas the data could find, and the validation footer that let a vice president believe four groups were worth four budgets. The statistics in this chapter are learnable in a week; the habit that distinguishes careers is refusing to call any partition a segmentation until a treatment, a definition, and a validation are attached — because a segmentation succeeds when the marketing calendar changes, and by no other test.

6.16 Ethics, Proxy Discrimination and Differential Treatment

The Business Analytics in Practice section described segmentation as the analytics that changes calendars. This section examines the ethical weight of exactly that power, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), and honest summarization (Section 5.14) — to the act this chapter adds: sorting people into groups in order to treat them differently. Differential treatment is not a side effect of segmentation; Section 6.2 defined it as the point. That is why this chapter's ethics cannot be a caution appended to a technique. The technique is the ethical object.

Begin with proxies, because clustering manufactures them by construction. StyleCraft's schema contains no race, religion, health, or income column, and it is tempting to conclude that its segmentations therefore cannot discriminate on such attributes. The conclusion is false, and the mechanism deserves to be understood rather than feared. Attributes are correlated: geography encodes income and, in most American metros, race; age band encodes life stage; channel behavior and device ownership encode affluence and age at once. A segmentation built on innocent columns can therefore sort customers by a sensitive attribute no column names — the disparate-impact

mechanism that Barocas and Selbst (2016) show can arise unintentionally, through historical data, inherited patterns, and proxies, rather than only through malice, and that Section 1.13 introduced as a standing concern of responsible analytics. It is a well-documented risk in data-driven decision systems, and treating it as one is the difference between an audit and an alibi.

Two consequences follow for practice. Dropping a sensitive column is not a defense: the information usually survives in its correlates, so “we never used age” and “our treatment does not vary by age” are separate empirical claims, and only the second matters. And the first instrument of the audit is the one this chapter already built for interpretability: holdout profiling. Profile every deployed segmentation on the sensitive and proxy attributes available — age band, gender, metro, region — precisely because they were not inputs, and read the result as an impact statement: who, demographically, lands in the segment that gets the richest offers, and who lands in the one that gets suppression? Section 6.10's validation tool and this section's fairness tool are the same instrument pointed at different questions.

Being the same instrument, it inherits the same limits, and this is where an ethics section must be more careful than a methods section. A profile of who lands where is a description of exposure, not a determination of fairness. It does not establish that a difference in treatment is unjustified, does not measure the benefit or burden the treatment actually delivers, does not settle whether a practice is lawful in any particular jurisdiction, and does not survive the moment the segmentation changes. Passing a demographic profile is a minimum screen — the first thing an analyst does, not the last thing an organization needs. The fuller screen adds four things: treatment rates computed by demographic and proxy group, not merely segment membership; an explicit comparison of the benefits and the burdens each group receives, since a segment that gets fewer discounts and more service is not obviously disadvantaged; a review of the eligibility and exclusion rules themselves, because the sharpest harms usually live in who is suppressed rather than in who is targeted; monitoring after deployment, because assignment shares drift; and an escalation path — to legal, to policy, to whoever owns the risk — wherever differential treatment creates material harm or touches a regulated domain. An analyst who validates properly has built the first instrument of that screen and none of the rest.

Differential pricing and offers are where segment-based treatment meets its sharpest public line. StyleCraft's catalog prices are uniform — a designed fact — but a CRM program does not need price tags to price-discriminate: allocating discounts, early access, free shipping, and win-back incentives by segment sets different effective prices for different people, and Section 6.11's treatment logic openly recommends it (“reward Champions, do not discount them; incentivize At Risk”). Much of this is legitimate, long-practiced, and welfare-defensible; some of it curdles, and the line has recognizable markers. Treatment keyed to behavior the customer controls and can understand — buy more often, earn better treatment — reads as a bargain and survives disclosure. Treatment keyed to who the customer is, or to proxies for it — the store-only suburban shopper quietly excluded from every promotion because her segment's profile says she pays full price — starts to price the person rather than the behavior, and its test is the one Section 2.12's framing ethics proposed: would the program survive being explained, in plain language, to the customers in its least-favored cell? A treatment map that cannot be read aloud is not a strategy; it is a secret.

The chapter's final caution is the one its own dataset was built to teach, and Section 6.14 has already sprung half of it. The New/At-Risk segment — short tenure, single purchases, unknown loyalty for many of its members — is measurably real, and it concentrates geographically where StyleCraft is newest, including the Wave 4 expansion markets. A naive value-based treatment policy deprioritizes that segment; a naive read of the early response data that follows goes further and concludes those markets underperform. The dataset's ground truth documents what the naive analysis misses: the apparent underperformance is an artifact of store age and marketing exposure, not of customer potential — new customers in under-marketed markets behave exactly as new customers in under-marketed markets should. This chapter is the first place an analyst can step into that trap, by letting a tenure artifact wear a value costume in a treatment map; the discipline that catches it here — same-age comparison, intensity reported beside level, tenure held out of the inputs and profiled afterward, and treatment decisions that name what a segment measures rather than what its averages resemble — is the discipline that catches its sharper versions when models start making the recommendations.

CONCEPT

Segments Treat People Differently — Audit Whom

Before any segmentation is deployed to vary treatment, run the minimum screen. Profile it on the sensitive and proxy attributes that were not inputs, and write down who receives the best and worst treatment in demographic terms. State, for each segment, whether its defining signal is behavior the customer controls or circumstance the customer occupies — tenure, geography, store age — and refuse value verdicts on circumstance. Apply the read-aloud test to the treatment map.

Then treat the screen as the floor it is. Failing the first check is disparate impact unexamined; failing the second is punishing people for when and where they arrived; failing the third is a pricing secret waiting for a screenshot. Passing all three establishes only that the obvious failures are absent. What a fairness assessment additionally requires — treatment rates by group, benefits weighed against burdens, exclusion rules reviewed, monitoring after launch, and escalation where harm or regulation is in play — is the organization's obligation, and the analyst's job is to say so rather than to let one profile table stand in for it.

Source: Course concept developed for this guide, informed by Barocas and Selbst (2016).

6.17 Chapter Summary

This chapter opened Part II by converting Chapter 5's insistence — these customers are not one population — into named, verified, targetable groups, and it equipped the CRM manager's four-slot decision with a defensible map. The main point is that segmentation is a treatment decision before it is a computation: a partition of customers becomes a segmentation only when each group is attached to a difference in treatment the organization can execute, and the organization's treatment capacity is a design input to the analysis, not an inconvenience discovered at deployment.

The chapter's machinery arrived in two movements. The strategic movement placed segmentation inside the segmentation-targeting-positioning framework, surveyed the four base families and the data each requires — behavioral bases dominating customer-level work because they are computed from verified acts — and separated the two roads to segments: a priori schemes declared in advance, transparent and instantly operable, and post hoc schemes estimated from data, capable of surprise and correspondingly burdened with proof. The analytic movement walked both roads. RFM analysis scored recency, frequency, and monetary value — frequency on declared bands rather than quintiles, because real frequency is lumpy and a row-order tiebreaker manufactures distinctions with no behavioral content — and arranged an exhaustive twenty-five-cell grid whose regions are journeys in waiting. The customer-grain feature table was built as a disciplined grain change on a declared purchasing population — with the customers outside it named as no purchase in the window rather than never purchased, and split between activation and win-back — reconciled to certified totals, with level features joined by exposure-normalized intensity features so that lifetime totals stop standing in for tenure unchecked, and with profiling attributes deliberately held out.

Distance made “alike” mathematical, standardization kept the mathematics honest — unscaled clustering is spend banding by accident — and the section closed on the limit of standardization: equalizing scale does nothing about redundancy, which is why feature selection is a modeling decision and the labs test it. k-means was developed to working level as an unsupervised method, with the assign-and-update sketch, initialization sensitivity, and the box worth carrying for a career: the algorithm returns the k you asked for, so structure is established by everything around the algorithm, never by its output. The choice of k braided elbow, silhouette, and business judgment; profiling translated centroids into evidence; and validation supplied the chapter's verification theme in two instruments with two different warranties — stability under reseeding and subsampling, measured with a relabeling-invariant index, which establishes reproducibility and not meaning, and holdout profiling, which is convergent evidence and not proof of natural kinds or of differential response. The last step was the one production forgets: the

fitted scaler, the fitted centroids, the feature definitions, and the population rule are retained and versioned together, because a segmentation nobody can rerun is a slide rather than a system.

The usefulness screen and a worked attractiveness matrix turned verified segments into targeting decisions, with differentiability scored as an untested hypothesis and substantiality restated as a burden of proof rather than a headcount test. The AI section named the assistant-era failure modes — the confident k, the skipped scaling, the invented persona, the single-run solution, the arbitrary tie rule — and installed the audit routine that answers them. The ethics section confronted the chapter's own purpose: proxies screened by the same holdout profiling that supports interpretation, offer allocation tested by whether it prices behavior or persons, the demographic profile named as a minimum screen rather than a fairness certification, and the tenure trap named before it could cost the newest markets their journey. The labs ran the full sequence on StyleCraft's designed data, where four planted segments graded the work.

Looking ahead, the treatment map describes who: four segments, sized, profiled, validated, and paired with journeys. The first question the map cannot answer is already on the table from the sign-off meeting — what moves these customers? Which levers, at what strength, with what confidence? Segments describe who; drivers explain what moves them. The next chapter takes up that question with the analyst's next instrument, regression — measuring how spend and behavior relate to engagement, tenure, discounts, and segment membership on the very feature table this chapter built — and with a new verification discipline to match, because the era of auditing AI-generated models has only begun.

6.18 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: attach a treatment to every group, choose bases the data can support, declare scoring rules including ties, scale before measuring distance, defend k on three legs, and validate every estimated structure by refit and holdout before naming it. They are organized into two groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Assignment #2, Segmentation and Targeting, draws on the homework sets. In-class activities are prepared for discussion rather than submitted. Each exercise also carries its assignment label — required practice, homework submission, or in-class discussion — so that instructors can assign selectively.

6.18.1 Core Chapter Practice

Exercise 6.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- State the difference between segmentation as a strategic act and segmentation as a computation, and give the opening case's example of each.
- Why is a set of personas, however vivid, not a segmentation as delivered? What single addition would convert one?
- Define a priori and post hoc segmentation, then state precisely what a post hoc method estimates from the data and what the analyst must still supply. Classify: the RFM grid; the intern's nine clusters; loyalty tiers; the labs' final deliverable (careful — it travels both roads).
- Quintile scores are often called self-calibrating. State what they preserve and what they do not, and give one comparison they cannot support without additional information.
- Why does k-means require standardized features? What does the clustering return if they are skipped, and why does standardization not solve the separate problem of redundant features?
- The elbow and the silhouette are both computed from the same fitted models. What different question does each answer, and what can neither answer?
- State the two validation instruments of Section 6.10, what each establishes, and — precisely — what each does not.

List the five useful-segment criteria, identify which one the platform's four-journey limit invokes, and explain why a fresh segmentation cannot honestly score the differentiable criterion as satisfied.

Name the five artifacts a deployed segmentation must retain, and explain what breaks if a team keeps only the written threshold rules.

Exercise 6.2 Pick the Base (Required Practice)

For each StyleCraft decision below, recommend a segmentation base family, name the specific columns that would operationalize it (or state that the schema cannot support it), and note one validity concern per Chapter 3.

Choosing which two metros receive a direct-mail catalog test.

Deciding which customers receive the win-back incentive.

Briefing the agency's creative team on the voice of the fall campaign.

Selecting customers for early access to the holiday occasionwear drop.

Reporting to the board on “the affluent customer opportunity.”

Exercise 6.3 Score the Grid by Hand (Required Practice)

Using only Lab 6.1 Part A's printed feature table (no code), assign every one of the ten miniature customers to a region of Table 6.3's grid, applying the declared frequency bands. Then answer three questions. Which customer's placement most misrepresents their likely value to StyleCraft, and which feature that the grid cannot see explains why? Table 6.3 has twenty-five cells and Table 6.4 lists seven regions — show that the mapping is exhaustive and mutually exclusive by counting the cells each region claims. Finally, write the frequency scoring rule you would document at full scale, in one sentence precise enough that two analysts applying it independently would produce identical scores, and state in one further sentence why rank(method="first") would not meet that standard.

Exercise 6.4 Audit the Scaling and the Feature Set (Required Practice)

A colleague clusters the full feature table without standardization and reports four segments. Using Section 6.7's worked trio (C002, C005, C006), show with hand arithmetic what fraction of the C002-C005 squared distance the monetary feature contributes, and state in one sentence what the colleague's four segments almost certainly are. Describe the one-line check — no refitting required — that would confirm your diagnosis from the colleague's own centroid table. Then extend the audit: the colleague standardizes and refits, and reports that the problem is solved. Name the two features in Table 6.5 that are mathematically dependent on others in the set, explain in two sentences why standardization does not address that, and state which cell of Lab 6.2 would settle the question and what result would justify keeping the eight-feature specification.

Exercise 6.5 Build, Cluster, and Map End-to-End (Homework Submission)

Complete Lab 6.1 Part B and Lab 6.2 on the certified files. Submit: the reconciled feature table with its written population rule and the three population counts; the full-scale RFM grid with the documented frequency rule; the evaluation sweep with a written three-leg defense of your chosen k and the candidates you rejected; the profiled and named segments with written assignment rules; all three validation results — seeded and subsample stability, feature-set sensitivity, and the holdout profile with your pre-written predictions — stated in plain sentences; the answer-key comparison; the completed attractiveness matrix with its evidence-versus-judgment column; the saved segmentation specification and the measured agreement of your simplified rule with it; and the one-page treatment map with its validation footer. Submissions are graded on the written rules, the k defense, the honesty of the validation footer, and the treatment map as heavily as on the code.

Exercise 6.6 AI-Assisted Cluster Audit (Homework Submission)

Give an AI assistant the feature table and this deliberately underspecified prompt, verbatim: “Segment these customers and describe the segments.” Then audit the response using Section 6.12's routine. Verify one centroid by hand. Refit its solution from three different seeds and report adjusted Rand indices rather than dominant-cell counts, explaining in one sentence why the index is the safer measure. Run the holdout profile. Check whether it scaled the features, how it handled tied frequency values if it scored RFM at all, and whether it retained the fitted scaler and

model or only described thresholds. List every attribute its segment names or descriptions asserted that no input feature or holdout column contains. Report what it chose for k and whether it disclosed choosing. Document the full exchange per the AI-use documentation template in Appendix D, and conclude with two sentences on which audit step changed your assessment most.

6.18.2 In-Class Activities

Exercise 6.7 Find the Flaw in the Segmentation (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

- An analyst clusters on recency_days, frequency, monetary, and tenure_days, and reports a “high-value veterans” segment and a “low-value newcomers” segment.
- A deck reports “the silhouette identified two segments,” and the fall program is built on urban versus suburban alone.
- A segmentation's overall average order value is reported as the simple average of the four segments' AOVs.
- An agency deliverable names a cluster “Eco-Conscious Millennials” from a table containing R, F, M, and channel shares.
- A team refits last quarter's clustering on this quarter's data, gets visibly different groups, and concludes the customer base has changed.
- The nine-cluster solution is defended in the meeting with “we validated it — the fit converged and the inertia is low.”
- A stability report cross-tabulates two runs, counts the customers outside each row's largest cell, and reports 96 percent stability. Two of the baseline clusters have their largest overlap with the same cluster in the refit.
- A campaign team is handed a deck listing four segments with threshold rules and asks how to assign the customers who signed up last week.

Exercise 6.8 Differential Treatment Mini-Cases (In-Class Discussion)

For each mini-case, apply Section 6.16's minimum screen — proxy profile, behavior versus circumstance, read-aloud test — then say which additional elements of the fuller screen the case requires, and come prepared to defend a verdict: proceed, modify, or refuse.

- The treatment map excludes the Suburban Occasion segment from all promotional emails “because they pay full price anyway.”
- A proposed “high-potential” journey is keyed to app ownership, and the holdout profile shows app ownership concentrates sharply in the two youngest age bands.
- Finance proposes suppressing all marketing to the New/At-Risk segment for two quarters to “let the low performers wash out,” with the savings funding a Champions expansion.
- A win-back incentive ladder offers the richest discounts to the highest-monetary lapsed customers, and the read-aloud test is applied from the perspective of a loyal modest-basket customer who receives the smallest offer.
- A segmentation passes its demographic profile with no visible imbalance, and the team proposes to record it as “fairness verified” in the release documentation.

6.19 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

A priori segmentation. Segmentation in which the number of segments and the assignment rule are declared before analysis, so that data serve to size and profile predefined groups (adapted from Wind, 1978).

Adjusted Rand index. A relabeling-invariant measure of agreement between two partitions of the same observations, computed over all pairs and corrected for chance agreement; 1.0 indicates identical partitions, values near 0 indicate unrelated ones, and negative values — bounded below at −0.5 — indicate partitions agreeing less

than chance would predict. Used in this chapter to measure clustering stability and feature-set sensitivity (adapted from Hubert & Arabie, 1985).

Cluster profiling. The description of algorithmically estimated groups — sizes first, centroids read in standardized and native units, spreads and medians alongside means — performed with the profile disciplines of Section 5.6 and preceding any naming of segments.

Customer-grain feature table. The analytic asset with one row per customer in a declared population and one column per feature, built from certified transaction data by written derivation rules at a declared analysis date and window, and reconciled to certified totals before downstream use.

Elbow method. The selection heuristic that plots inertia against candidate values of k and nominates the bend beyond which additional clusters buy sharply diminishing compactness (adapted from Thorndike, 1953).

Euclidean distance. The straight-line distance between customers represented as feature vectors — the square root of the sum of squared feature differences — whose value depends on every feature's units and spread (adapted from Everitt et al., 2011).

Exposure normalization. The expression of a level feature as a rate over the time a customer was actually observed inside the analysis window, reported beside the level feature so that a lifetime total bounded by tenure is not read as a measure of customer value. It reduces the mechanical tenure effect rather than removing it, since the level features remain in the specification.

Inertia. The total within-cluster sum of squared distances from observations to their assigned centroids; the quantity k-means minimizes. At the optimum it is non-increasing as k rises, so it informs the choice of k only through the shape of the curve, and a reported increase is a signal to inspect initialization and convergence.

k-means clustering. The unsupervised partitioning algorithm that locates k centroids and assigns each observation to its nearest centroid to minimize inertia, fitted by alternating assignment and update steps from an initialization on which the solution can depend (adapted from MacQueen, 1967; Lloyd, 1982; Pedregosa et al., 2011).

Market segmentation. The division of a market into internally similar, externally different groups of buyers so that marketing effort can be differentiated across groups rather than uniform over the whole market (adapted from Smith, 1956; Kotler & Keller, 2016).

Post hoc segmentation. Segmentation in which group membership and boundaries are estimated from the data by an analytic method such as clustering, after the analyst specifies the features, the method, and the candidate values of k, and carrying a corresponding burden of validation (adapted from Wind, 1978).

Quintile scoring. The conversion of a numeric quantity into a five-level ordinal score by ranking customers and assigning 5 to the marketing-favorable fifth through 1 to the least; scores preserve rank within the scored population and window but not absolute level, and tie handling is declared as part of the rule (adapted from Hughes, 1994).

RFM analysis. The behavioral segmentation method that scores customers on recency, frequency, and monetary value in a declared window and groups them by score combinations on an exhaustive named grid, with each region's treatment logic treated as a hypothesis about response rather than an established rule (adapted from Hughes, 1994; Fader et al., 2005).

Segment attractiveness. The comparative evaluation of qualifying segments on size, value, growth, margin, cost to reach, and strategic and capability fit, used to inform targeting priority, with measured cells marked separately from judgment cells (adapted from Kotler & Keller, 2016; Wedel & Kamakura, 2000).

Segmentation base. The characteristic or set of characteristics — geographic, demographic, psychographic, or behavioral — on which customers are divided, determining the data required and the treatments supportable (adapted from Wedel & Kamakura, 2000; Kotler & Keller, 2016).

Silhouette score. A per-observation comparison of average within-cluster distance against average distance to the nearest other cluster, scaled to lie between −1 and +1 and averaged across observations to grade a clustering's geometric separation at each candidate k (adapted from Rousseeuw, 1987).

Stability check. The validation instrument that refits a clustering under perturbation — independently seeded initializations, subsamples — and measures agreement between the resulting partitions with a relabeling-invariant index; high agreement establishes reproducibility, not meaning (adapted from Everitt et al., 2011; Hubert & Arabie, 1985).

Standardization (z-score scaling). The rescaling of a numeric feature by subtracting its mean and dividing by its standard deviation, giving mean 0 and standard deviation 1 so that distances weight features in standard-deviation units rather than native units; it equalizes scale and does not remove redundancy (adapted from James et al., 2021; Pedregosa et al., 2011).

STP (segmentation, targeting, positioning). The sequential strategy framework of dividing a market into segments, selecting target segments, and positioning the offering distinctively for each target (adapted from Kotler & Keller, 2016).

Targeting. The selection of which segments to serve, with what priority and resources, decided on segment-attractiveness evidence within the firm's treatment capacity (adapted from Kotler & Keller, 2016).

Useful-segment criteria. The five-part screen — measurable, substantial, accessible, differentiable, actionable — that a segment must pass to justify differentiated marketing (adapted from Kotler & Keller, 2016).

6.20 Further Readings

Students who want additional background may begin with the following readings. Strategy and marketing foundations are listed first, methods second.

Smith (1956) for the founding statement of segmentation as strategy — short, readable, and the source of the idea every algorithm in this chapter serves.

Yankelovich and Meer (2006) for the sharpest practitioner critique of segmentation drift — how segmentations decay into demographic portraiture and how to key them to decisions instead; the extended companion to Sections 6.2 and 6.15.

Wedel and Kamakura (2000) for the comprehensive scholarly treatment of segmentation bases and methods; the reference behind Sections 6.3 and 6.11.

Fader et al. (2005) for the bridge from RFM measures to formal customer valuation — where Section 6.5's workhorse meets the customer-lifetime-value modeling of later chapters.

James et al. (2021), Chapter 12, for the standard accessible treatment of k-means and its relatives one level of formality above this chapter, read alongside Rousseeuw (1987) for the original silhouette paper — brief, geometric, and clarifying about what separation scores do and do not certify.

Barocas and Selbst (2016) for the definitive legal-analytic treatment of disparate impact in data-driven decisions; the foundation under Section 6.16, first cited in Section 1.13.

6.21 References

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 1027–1035). Society for Industrial and Applied Mathematics.

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (2011). *Cluster analysis* (5th ed.). Wiley.

Fader, P. S., Hardie, B. G. S., & Lee, K. L. (2005). RFM and CLV: Using iso-value curves for customer base analysis. *Journal of Marketing Research, 42*(4), 415–430. https://doi.org/10.1509/jmkr.2005.42.4.415

Haley, R. I. (1968). Benefit segmentation: A decision-oriented research tool. *Journal of Marketing, 32*(3), 30–35. https://doi.org/10.1177/002224296803200306

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218. https://doi.org/10.1007/BF01908075

Hughes, A. M. (1994). *Strategic database marketing*. Probus Publishing.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer.

Kotler, P., & Keller, K. L. (2016). *Marketing management* (15th ed.). Pearson Education.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory, 28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). University of California Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7

Smith, W. R. (1956). Product differentiation and market segmentation as alternative marketing strategies. *Journal of Marketing, 21*(1), 3–8. https://doi.org/10.1177/002224295602100102

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika, 18*(4), 267–276. https://doi.org/10.1007/BF02289263

Wedel, M., & Kamakura, W. A. (2000). *Market segmentation: Conceptual and methodological foundations* (2nd ed.). Kluwer Academic Publishers.

Wind, Y. (1978). Issues and advances in segmentation research. *Journal of Marketing Research, 15*(3), 317–337. https://doi.org/10.1177/002224377801500302

Yankelovich, D., & Meer, D. (2006). Rediscovering market segmentation. *Harvard Business Review, 84*(2), 122–131.

Chapter 07

Relationships, Drivers, and Regression

From Who to What Moves Them: Measuring, Modeling, and Refusing to Overclaim the Drivers of Customer Spend

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops the analysis of relationships among marketing variables: how association is measured, modeled, and kept distinct from causation. It builds linear association through covariance and the Pearson correlation coefficient, then catalogs what correlation hides — nonlinearity, outlier sensitivity, and mixed populations — and develops confounding, defined as a pre-existing common cause, on marketing's most reliable confounder, the calendar. Linear regression is built from the simple case through multiple regression, dummy variables, and interactions, with coefficients read as model-adjusted comparisons rather than levers and interactions read conditionally. Statistical evidence is developed at working level: heteroskedasticity-robust standard errors, confidence intervals, p-values, and variance inflation factors. A section on AI assistance installs the four-point audit of signs, magnitudes, units, and diagnostics. The labs estimate spend and average-order-value drivers on the Chapter 6 feature table, recover the chapter's designed interaction, and verify an AI-fitted model against certified descriptive findings.

KEYWORDS

correlation; covariance; regression; drivers; dummy variables; interaction terms; multicollinearity; confounding; omitted variable bias; causal overclaiming

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Relationships, drivers, and regression. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 7, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, NumPy, and Matplotlib are sponsored projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. statsmodels is a community-developed project distributed under the modified BSD license. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and

analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

26. Distinguish association from causation as a standing discipline, and explain why "driver" is a causal word that correlational evidence places on probation.
27. Compute and interpret sample covariance and the sample Pearson correlation coefficient among marketing variables, stating the direction, the strength, the unit-free character of r, and the difference between the population quantity and the sample estimate the software reports.
28. Identify what correlation misses — nonlinearity, outlier sensitivity, and mixed populations — and apply the always-plot discipline before trusting any correlation.
29. Read a correlation matrix without committing its characteristic misreadings, and explain why a matrix of pairwise correlations is not a map of drivers.
30. Define a confounder as a pre-existing common cause, distinguish it from a variable that is merely correlated with both members of a pair, and recognize seasonality as marketing's standing confounder.
31. Construct and interpret simple linear regression models: the fitted line, the intercept and slope in native units, least squares as a criterion, and residuals as diagnostic evidence.
32. Interpret R^2 as shared-variation fit under its stated conditions, and explain why fit and usefulness are different judgments, anchored to the baseline discipline of Section 2.7.
33. Construct and interpret multiple regression models, read a coefficient as a model-adjusted comparison rather than a lever, and reason about omitted variable bias and its direction, including the qualification that applies once a model holds several predictors.
34. Encode categorical variables with dummy variables and reference categories, interpret segment coefficients against the reference, and specify and interpret an interaction term conditionally.
35. Assess coefficient evidence using heteroskedasticity-robust standard errors, confidence intervals, p-values, and variance inflation factors at working level, and distinguish prediction from explanation as modeling goals.
36. Verify AI-generated regression output against fundamentals — signs, magnitudes, units, fit, and diagnostics — and against certified descriptive findings, documenting the audit per Appendix D.

Chapter 6 ended with a map and a question the map could not answer. The map was the treatment map: four segments, estimated, validated, named by rule, and paired with journeys on the fall calendar. The question came from the sign-off meeting, and it was the natural next one — what actually moves these customers? Which levers, at what strength, with what confidence? Segments describe who; this chapter is about what moves with their behavior, and about the analyst's principal instrument for that question: regression. The route runs from the raw material of relationships — association measured honestly by correlation, and the many ways correlation deceives — through the regression model itself, simple and then multiple, with the categorical machinery that lets StyleCraft's estimated segments enter the model as evidence. Because every step of this route is now one prompt away from an AI assistant, the chapter also installs its own verification discipline — the audit of signs, magnitudes, units, and diagnostics — and its own ethical guardrail, because the characteristic failure of this chapter's tools is not a wrong number but a wrong verb: an association, fluently narrated, wearing the language of cause.

CONCEPT

What This Chapter Is Really About

Chapter 6 argued that grouping is the third analysis, and that an algorithmic grouping is a hypothesis to be validated rather than a fact to be admired. This chapter takes the next step in the same argument: relating is the fourth analysis, and it is the first whose output looks like an answer to the question executives actually ask — what should we do? A correlation of 0.6, a coefficient of $41 per order, a fitted line through a cloud of customers: each is arithmetic about the world as it is, and each will be read, the moment it reaches a meeting, as a claim about what would happen if the world were changed. The entire discipline of this chapter lives in that gap.

The methods are not hard; a modern AI assistant fits a defensible regression in seconds. What cannot be delegated is the reading: whether the signs and magnitudes survive contact with certified facts, whether the model's assumptions survive contact with its residuals, and whether the sentence written about the coefficient claims only what the evidence can carry. An analyst who masters this chapter does not become the person who finds drivers. They become the person who knows exactly how much of "driver" the data has actually earned.

Source: Course concept developed for this guide, informed by Freedman (1991) and Shmueli (2010).

7.1 Marketing Decision Context: Three Levers and One Budget

Chapter 6 left the CRM manager in possession of a signed treatment map: four validated segments — the Urban Loyal Core, the Suburban Occasion shoppers, the Discount-Dependent, and the New/At-Risk — each attached to one of the four journeys the platform can sustain this fall. The map answered who. The sign-off meeting immediately generated the next decision, and this one has a budget attached. Beyond the fixed cost of building the four journeys, StyleCraft's fall retention program carries an incremental budget — the money that funds what the journeys actually do — and the VP of Marketing must allocate it across three candidate levers before the pre-holiday budget lock in twenty-one days. The three levers on the table: deepen the discount ladder (richer offers, triggered more often), accelerate app adoption (an install campaign plus app-exclusive perks), and increase email frequency for enrolled customers. The CFO will attend the allocation meeting. After the margin conversation that opened this guide in Chapter 1, every dollar of discount is presumed guilty until proven incremental.

Three claims are already circulating in the building, one per lever, and each arrives dressed in numbers. The app claim comes from the platform team's deck: "App users spend 2.4 times more than non-app customers. App adoption is our highest-leverage investment — every install is worth hundreds of dollars in annual spend." The discount claim comes from a finance analysis: "Customers with high discount exposure show below-average total spend. Discounts are training our customers to pay less — cut the ladder, and spend will recover." The email claim comes from the CRM team itself: "Email-engaged customers buy twice as often as the unengaged. Doubling send frequency doubles touches, and touches drive orders." Each claim cites a real pattern; each pattern will reappear, verifiable, in this chapter's labs. And each claim commits, in its second sentence, an inference its first sentence cannot support. The VP has asked the analyst — you — for something specific by the budget lock: an honest drivers analysis of customer spend. Which variables actually move with spend, how strongly, in what units, holding what constant — and, for each of the three claims, a verdict: what the data supports, what it cannot support, and what evidence would settle the difference.

The deliverable has a shape this chapter will fill in. First, the relationships themselves, measured honestly: which of StyleCraft's behavioral and engagement variables are associated with spend, at what strength, and with what caveats about shape, outliers, and the mixed population Chapter 6 showed the base to be. Second, a model: spend regressed on the candidate drivers together, so that each claim can be examined holding the others constant — because the app claim's "2.4 times" compares customers who differ in far more than app ownership, and the meeting

deserves to know how much of the gap survives when tenure, recency, engagement, and segment membership are held level. Third, the segments as evidence: the four groups from Chapter 6 enter the model as dummy variables, both because segment membership is itself a candidate driver and because Chapter 5's certified findings about the segments — the suburban average-order-value gap and its decomposition — are the known truths against which any fitted model must be checked. Fourth, and running through everything, the language: the deliverable's sentences must survive a CFO who has learned to ask, of every "driver," whether anyone actually changed anything and watched what happened.

The stakes of the language are not academic. If the app coefficient is read causally and the install campaign is funded on it, the budget buys the customers StyleCraft already has — the young, urban, loyalty-enrolled core who were always going to install the app — dressed up as growth. If the discount association is read causally and the ladder is cut, the Discount-Dependent segment, one quarter of the treatment map, loses the behavioral lever with which its purchases are most visibly associated, three weeks before the holiday, on evidence that has never tested whether the segment responds to that lever at all. The chapter builds the machinery in order. Sections 7.2 through 7.5 discipline the raw material: association and causation, correlation and its failure modes, confounding and the calendar. Sections 7.6 through 7.10 build the model: simple regression, fit, multiple regression, the categorical machinery, and the reading of statistical evidence. Section 7.11 turns to the AI assistant, which will fit all of this fluently and narrate it causally by default. The labs in Section 7.12 estimate the drivers of spend and average order value on the feature table Chapter 6 built — where, by design, the true coefficient structure is known — and Sections 7.13 through 7.15 rehearse the meeting: the managerial translation, the industry mirror, and the ethics of the verb "drive."

7.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

18. Each of the three lever claims moves from an observed difference to a recommended action. For each claim, state the comparison actually being made in the data, and name one way the compared groups could differ other than the lever itself.
19. The app claim's "2.4 times" is a ratio of group means. What additional facts about the two groups would you want before treating that ratio as even descriptively meaningful — and which chapter of this guide taught each fact?
20. The finance claim reasons from a negative association between discount exposure and spend. Chapter 6's segmentation gives a specific reason this association could exist even if discounts increase every individual customer's spending. State it.
21. The VP asks for "drivers." The CFO asks whether anyone changed anything and watched what happened. What kind of evidence would satisfy the CFO's standard, and why is this chapter's evidence not it?

7.2 Association, Causation, and the Meaning of "Driver"

Everything in this chapter rests on one distinction, and the guide has been preparing it since Chapter 2 named the anatomy of a decision. Two variables are associated when they move together: when knowing one tells you something about the likely value of the other. One variable causes another when intervening on the first changes the second — when the world after the intervention differs from the world without it. Association is a fact about observed data; causation is a claim about what would happen under an action not yet taken. Marketing decisions are actions, so marketing questions are, in their bones, causal — "should we deepen the discount ladder?" is a question about a world that does not yet exist. Marketing data, almost always, records associations. The gap between the question's nature and the data's nature is the standing condition of the marketing analyst's work, and this chapter's tools live entirely on the association side of the gap.

DEFINITION

Association and Causation

Two variables are associated when their values move together in observed data, so that knowing one improves a guess about the other. A variable causes another when changing the first, with all else undisturbed, would change the second. Association is symmetric and observational; causation is directional and interventional. Observed association is consistent with causation in either direction, with a third variable influencing both, or with coincidence — and cannot, by itself, distinguish among them.

Source: Adapted from Simon (1954) and Freedman (1991).

In other words: the data can show that app users spend more; it cannot, alone, show that the app made them do it. The definition's last sentence enumerates the rivals every association carries. The app–spend association could exist because the app increases spending (the platform team's reading); because spending increases app adoption — frequent customers install the app to track orders they were already going to place (reverse causation); because a third variable, engagement with the brand, influences both installs and orders (confounding, Section 7.5's subject); or, in small samples, because of nothing at all. Nothing in the number 2.4 favors any of these. Ruling rivals out requires either design — the experiments of Chapter 11, where random assignment severs every backdoor — or assumptions strong enough to substitute for design, which multiple regression will attempt in Section 7.8 and only partly achieve.

This is why the word "driver," the word the VP used and the word this chapter's title uses, deserves formal probation. In business language, a driver analysis is understood loosely: which variables move with the outcome strongly enough to matter. In causal language, a driver is a lever: push it and the outcome moves. The looseness is not harmless, because the loose sense is what the analysis delivers and the strict sense is what the budget meeting hears. This guide's convention, which the deliverable in Section 7.13 will follow, is to use "driver" only with its probation officer present: a driver, in this chapter, is a variable that predicts the outcome in a specified model, and every sentence that promotes it to a lever must say what additional evidence performed the promotion.

CONCEPT

"Driver" Is a Causal Word on Probation

The deliverable is called a drivers analysis; the evidence is associational. Manage the gap explicitly. In writing and in meetings, let "is associated with," "predicts," and "differs by" carry the findings, and reserve "drives," "lifts," "causes," and "will produce" for claims backed by intervention evidence or clearly labeled assumption. The discipline is not pedantry; it is the difference between the analysis the data performed and the analysis the meeting will remember. Section 7.15 examines what happens — and to whom — when the verbs drift.

Source: Course concept developed for this guide, informed by Freedman (1991).

None of this makes association worthless — the opposite. Association is evidence: it nominates candidate levers, sizes relationships, falsifies confident stories (a lever uncorrelated with the outcome has a steep hill to climb), and disciplines the prioritization of the expensive causal tests to come. Chapter 2's evidence discipline applies: the analyst's job is not to refuse conclusions but to match the confidence of the sentence to the strength of the evidence. The rest of this chapter builds the machinery for measuring association well; the standing distinction rides along in every section, and the ethics section will show it is the chapter's real subject.

7.3 Covariance and Correlation: Putting a Number on Linear Association

Measurement starts, as it did in Chapter 3, with a definition decision: what number should summarize "moving together"? The natural starting point is covariance. Take two variables at the customer grain — frequency and monetary value from the feature table Chapter 6 built. For each customer, measure how far each variable sits from its own mean, and multiply the two deviations. A customer above average on both, or below average on both, contributes a positive product; a customer above on one and below on the other contributes a negative product. Covariance summarizes these products across customers: positive when the variables tend to sit on the same side of their means, negative when they sit on opposite sides, near zero when there is no consistent pattern.

One detail in that summary decides whether the printed number will match the definition, and this guide fixes it now rather than letting the software fix it silently. There are two covariances, and they differ in their denominator. The population covariance is the mean product of paired deviations across a full population: sum the products, divide by the number of members. The sample covariance estimates that quantity from a sample by dividing the sum of products by one fewer than the number of observations. The distinction is not decorative here, because the labs run on pandas, and pandas reports the sample version by default — so a chapter that defines covariance as "the average product" and then prints a number computed with the smaller denominator has taught two different quantities under one name.

DEFINITION

Covariance

The population covariance of two variables is the mean product of their paired deviations from their respective means, positive when the variables tend to deviate in the same direction and negative when they tend to deviate in opposite directions. The sample covariance estimates it from n observations by dividing the sum of the paired deviation products by n − 1. Its magnitude depends on the units of both variables, which makes it a direction detector rather than a strength measure. In this chapter, pandas reports the sample covariance and the sample correlation unless stated otherwise.

Source: Adapted from Rodgers and Nicewander (1988).

Covariance has the right sign and the wrong scale, and the reason echoes Section 6.7's tyranny of units: the products are denominated in the units of both variables at once — order-dollars, in the frequency–monetary case — so the number's size says as much about measurement units as about the relationship. Chapter 6 solved this problem for distance by standardization, and correlation is the same solution applied to covariance: divide by the spread of each variable. The result is the Pearson correlation coefficient, written r, and the division buys three properties that made it the lingua franca of association. It is unit-free: r between frequency and monetary is the same whether monetary is in dollars or cents. It is bounded: r always lies between −1 and +1, with the endpoints reached only when the points fall exactly on a line. And it is symmetric: r of X with Y equals r of Y with X — a property to remember, because "drivers" language is directional and r is not.

DEFINITION

Pearson Correlation Coefficient (r)

The sample Pearson correlation coefficient is the sample covariance of two variables divided by the product of their two sample standard deviations. Because the same n − 1 denominator appears in the covariance and in each standard deviation, the correlation is unaffected by the choice of denominator even though the covariance is not. It measures the direction and strength of linear association on a fixed, unit-free scale from −1 (perfect negative linear association) through 0 (no linear association) to +1 (perfect positive linear association).

Source: Adapted from Pearson (1895) and Rodgers and Nicewander (1988).

That last property is worth one further sentence, because it explains why the sample-versus-population distinction matters for covariance and not for r. Dividing a covariance by two standard deviations cancels the denominator: whichever convention is used, provided it is used consistently in all three quantities, the ratio is identical. So the covariance a student computes by hand can disagree with pandas in the third digit while the correlation agrees exactly — which is itself a small, memorable lesson about where conventions bite and where they do not, and Lab 7.1 arranges for the student to meet it.

Reading r takes calibration, and Table 7.1 supplies the working scale this guide uses — with the warning that context outranks any fixed ladder. In customer-level marketing data, where individual behavior is noisy by nature, correlations above 0.5 between distinct behaviors are strong findings and correlations near 0.9 should trigger suspicion rather than celebration: behaviorally distinct quantities rarely track each other that tightly, and near-perfect r usually means the two columns are the same fact wearing two names — monetary and its own components, a metric and its numerator, the arithmetic relatives Section 7.10 will flag for the model. The table is introduced here and consulted throughout the labs.

Table 7.1

Reading r in customer-level marketing data

Range of |r|	Working label	Reading in this guide	Standing caution
0.0–0.1	Negligible	No linear association worth a sentence	Could still hide a nonlinear pattern (Section 7.4)
0.1–0.3	Weak	Detectable; matters mainly in aggregate	Easily produced by a shared confounder
0.3–0.5	Moderate	A real behavioral relationship	Plot before believing; check outlier influence
0.5–0.8	Strong	Unusually tight for distinct behaviors	Ask what shared machinery produces it
0.8–1.0	Suspect	Rare between genuinely distinct quantities	Often two names for one fact; check derivations

Three boundary facts complete the working picture. First, r is undefined when either variable has zero variance: the formula divides by both standard deviations, so a variable that never moves has no correlation with anything, and software will report NaN rather than zero. The practical form of this rule is a warning about near-misses — inside a narrow subgroup, a variable with almost no spread yields an r that is numerically defined and substantively meaningless, which Lab 7.1's within-segment exercise is built to expose. Second, r is a customer-grain statistic here, and the grain declaration discipline of Section 3.3 applies with full force: the correlation between frequency and monetary across customers is a different fact from the correlation between daily orders and daily revenue across days, and Section 7.5 will show that the daily version imports the calendar as a hidden third variable. Third, r is a two-variable statistic, and modern practice rarely stops at one pair: the correlation matrix — every pairwise r among a set of variables, arranged in a grid — is the standard first exhibit of a drivers analysis, and it is also the exhibit most reliably misread. The matrix's seductions are specific enough to deserve their own treatment, and they open the next section.

7.4 What Correlation Misses: The Always-Plot Discipline

The correlation coefficient compresses a full scatter of customers into one number, and compression loses information by design. This section catalogs what is lost — and installs the discipline that recovers it, which costs one line of code and has been part of this guide's toolkit since Chapter 4's exploratory charts: plot the data before, or at latest alongside, any correlation you intend to repeat in a meeting.

The canonical demonstration is Anscombe's quartet: four small datasets constructed so that the means, variances, correlation (r = 0.816), and fitted regression line are essentially identical across all four — while the scatterplots show four utterly different worlds: a well-behaved linear cloud; a clean curve that is not a line at all; a

perfect line with one outlier that has dragged the fitted slope; and a pillar of identical x-values propped into a "relationship" by a single distant point (Anscombe, 1973). Anscombe built the quartet to argue against the habit of computing without graphing, and half a century later the argument has sharpened, because the computing is now delegated: an AI assistant asked for a correlation matrix returns the numbers, not the shapes, and narrates the numbers fluently. The quartet's lesson, restated for this guide: identical statistics do not mean identical evidence, and the scatterplot is the cheapest audit in analytics.

CONCEPT

Always Plot — the Anscombe Discipline

No correlation, and later no regression, enters a deliverable without its scatterplot having been seen by the analyst. The plot answers four questions r cannot: Is the relationship linear, so that r is even the right summary? Is it produced by a handful of extreme customers? Is it one relationship, or several populations overlaid? Is there any pattern at all in a region that matters — or is the association concentrated where the business isn't? One glance answers all four. The discipline is mechanical, it is the chart-reading skill of Chapter 4 pointed at a new target, and its absence is detectable in a deliverable the way a missing verification log was in Chapter 4: confident numbers, no evidence anyone looked.

Source: Course concept developed for this guide, informed by Anscombe (1973).

Three specific losses matter most on StyleCraft's data, and each is a designed feature of the base the labs will meet. Nonlinearity first: r measures linear association only, and marketing relationships are routinely curved. Recency and future spending, engagement and response, tenure and frequency — each tends to rise or fall steeply in one region and flatten in another, and a straight-line summary of a curve understates the relationship where it is strong and overstates it where it is flat. An r near zero, in particular, does not mean "no relationship"; it means "no linear one," and the plot decides which.

Outlier sensitivity second: r is built from products of deviations, so a single customer far from both means contributes an enormous product — Chapter 5's concentration analysis showed that StyleCraft's top decile holds an outsized share of revenue, and those whales can single-handedly manufacture, inflate, or reverse a correlation among spend variables. The lab's routine, executed in Code 7.5: compute r with and without the top spend decile, and if the story changes, the story was about the whales.

Third, and most important on this dataset: mixed populations. Chapter 6 established that the base contains four segments with different behavioral regimes. A correlation computed across the pooled base averages over regimes that may individually show different relationships — frequency and basket size can be positively related within every segment and negatively related across the pool, because the pool's variation is dominated by the between-segment contrast of frequent small-basket urban customers against infrequent large-basket suburban ones. This is the correlational face of the aggregation lesson of Section 3.3.1 — the claim changes with the grain, and here, with the population sliced — and the working rule is the one Chapter 5 used for every summary: compute the pooled number, then compute it within segments, and treat disagreement between the two as a finding about structure, not a nuisance.

The correlation matrix concentrates all of these hazards in one exhibit, plus two of its own. Table 7.2 collects the misreadings this guide's students most reliably commit on their first matrix, with the corrections the sections of this chapter supply. The matrix remains the right first exhibit — it is the fastest wide-angle view of a feature table — but it is a screening instrument, and Table 7.2's right-hand column is the protocol for everything it flags.

Table 7.2

Correlation-matrix misreadings and their corrections

Misreading	Why it is wrong	Correction
"The biggest r is the biggest driver"	r is symmetric association, not directed influence; and the biggest r is often an arithmetic relative	Rank candidates only after Section 7.8's model adjusts for the others; check derivations first
"r ≈ 0, so this variable doesn't matter"	Zero linear association can hide a strong curve or offsetting subgroup patterns	Plot it; compute within segments (Section 7.4)
"These two drivers both correlate with spend, so both add value"	They may carry the same information — multicollinearity in waiting	Check their correlation with each other and their variance inflation factors; Section 7.10
"r = 0.9 — excellent, strong finding"	Near-perfect r between distinct behaviors usually flags a definitional overlap	Trace both columns' derivation rules (Table 6.5) before celebrating
"The matrix shows what to change to move spend"	Every cell is observational; no cell survives Section 7.2's rivals	Treat the matrix as a nomination list for modeling and, later, testing

One habit closes the section, borrowed whole from Chapter 1's workflow: predict before you compute. Before running the matrix in Lab 7.1, the analyst writes the expected sign of every correlation with spend — recency negative, engagement positive, and so on — from marketing logic and from Chapter 5's certified findings. The matrix then grades the predictions, and every surprise is an assignment: a plot, a derivation check, or a within-segment recomputation. Surprises that survive the assignment are findings. This is predict-then-verify (Section 1.7) applied to association, and it is the small-scale rehearsal of the audit this chapter's AI section will formalize.

7.5 Confounding and Spurious Correlation

Section 7.2 listed the rivals behind every association; this section develops the one that does the most damage in marketing, because marketing's operating environment manufactures it on a schedule. An association between X and Y is confounded when a third variable Z, present before either, influences both — creating or distorting a correlation between X and Y that partly or wholly reflects Z's double influence rather than any relationship between X and Y themselves. When the observed association has no direct component at all — when it exists only through Z or through coincidence — it is called spurious: real as a number, empty as evidence (Simon, 1954).

The definition has to be tighter than the one students usually carry, and the looseness is worth naming because it produces real errors. "A variable related to both" is too broad a test. A variable that sits on the pathway between X and Y — a mediator — is related to both, and adjusting for it removes exactly the part of the relationship the analysis was trying to see. A variable that is a common consequence of both — a collider — is also related to both, and adjusting for it can manufacture an association where none existed. This chapter does not develop causal diagrams; at working level the repair is to define the confounder by its position rather than by its correlations, and to treat a double correlation as a flag that starts an inquiry rather than as a verdict that ends one.

DEFINITION

Confounding and Spurious Correlation

A confounder is a pre-treatment variable that influences, or helps determine, both the predictor or exposure and the outcome, creating or distorting their observed association. Being correlated with both variables is a warning sign, not by itself proof that a variable is a confounder: a variable on the pathway between them, or a common consequence of them, is also correlated with both and calls for different handling. An association is spurious when it reflects only the operation of confounders or chance, so that neither variable responds to the other. Detecting confounding requires variables and reasoning beyond the correlated pair itself; nothing in the pair's own correlation reveals it.

Source: Adapted from Simon (1954) and Freedman (1991).

The definition's closing sentence is the operational point: the correlation between X and Y, examined ever more closely, will never confess that Z exists. Confounders are found by thinking about the world, not by staring at the pair — which is why domain knowledge is a technical input to this chapter, not decoration around it. Public collections of absurd examples make the logic memorable — time-series pairs like per-capita cheese consumption tracking engineering doctorates, correlated at 0.9 and connected by nothing but parallel drift (Vigen, 2015) — but the absurd cases are the safe ones, because no one budgets on cheese. The dangerous confounders are the plausible ones, and marketing's most reliable confounder is the calendar.

Here is the seasonality example this guide will reuse for the rest of Part II. Take StyleCraft's daily grain: marketing activity by day and revenue by day, across the 24-month window. Email volume and revenue will correlate strongly and positively across days. The mechanism is only partly the one the email claim wants: StyleCraft, like every retailer, concentrates its marketing where demand already concentrates — November and December, the designed holiday lift of the dataset's own revenue curve, plus the back-to-school swell. The calendar raises sends (campaign planners schedule into the season) and raises revenue (customers buy for the holidays regardless of email), and the daily correlation gratefully records the two effects as one. A naive read — "days with more email have more revenue; email drives revenue; send more" — is the confounded read, and its recommendation self-destructs in February. The correction pattern generalizes far beyond email: compare within season rather than across it (does extra email correlate with extra revenue among December days?), bring the confounder into the analysis as a variable — which is precisely the job Section 7.8's multiple regression takes up — or change the design so the confounder cannot operate, which is Chapter 11's business. Note what the example does to the email claim from the opening case: the claim's customer-grain version (engaged customers buy more often) and its day-grain version (heavy-send days sell more) are different claims with different confounders, and neither, unexamined, supports the recommendation to double frequency.

Confounding also explains the finance claim's trap, and Chapter 6 already built the machinery to see it. Discount exposure and total spend correlate negatively across the pooled base — the pattern is real and the labs will confirm it. But segment membership is associated with both discount exposure and total spend: the Discount-Dependent segment concentrates its purchases on promotion and spends less in total than the Urban Loyal Core, while the Core buys at full price and spends the most. The pooled correlation records that composition as if it were a consequence — the same between-group mechanism as Section 7.4's mixed-population warning. The segment label earns its place here as a descriptive stratifier, because it exposes the customer composition sitting behind the pooled association. It does not automatically earn the stronger title. Chapter 6 estimated that label from same-window behavioral features that include monetary value and discount share — the very outcome and the very exposure this example is examining — so the label is not an independently measured pre-treatment variable. It is partly constructed from the behaviors whose relationship is under study, and treating it as a clean confounder or a causal control would be reasoning in a circle: controlling for a grouping that was defined, in part, by the exposure and the outcome themselves. This is worth stating plainly because the reflex it corrects is common. An analyst who has just built a segmentation naturally reaches for it as the obvious control, and the reach is right about the arithmetic and wrong about the warrant.

What the segment label does support is a specific and useful list: exposing the composition behind a pooled number, describing heterogeneity, testing whether pooled and within-group patterns disagree, reproducing certified descriptive structure with a new instrument, and improving an explanatory model's descriptive fit. Whether discounts reduce, leave unchanged, or increase any individual customer's spending is simply not answered by the pooled r, and it is not answered by the stratified version either. The within-segment recomputation diagnoses how composition shapes the pooled association; the fitted model with segment dummies (Section 7.9) is a second descriptive adjustment, useful for examining heterogeneity and insufficient for causal identification. A genuine pre-treatment confounder — the calendar, tenure at the time of exposure, a demographic measured at signup — is a different kind of variable, and Section 7.8 returns to what controlling for one can and cannot buy. The finance claim may still be right. The point of this section is that its current evidence cannot tell.

What correlation cannot do, then, is settle any of the opening case's three claims. What it has done, by the end of Lab 7.1, is discipline them: signs and strengths measured, shapes inspected, whales counted, segments separated, and the calendar named as a standing suspect. The next step is the instrument that examines relationships jointly instead of pairwise — and whose output format, the coefficient, is the one the meeting will actually quote.

7.6 Simple Linear Regression: The Line and Its Coefficients

Correlation answers "do these move together, and how tightly?" The budget meeting needs a different quantity: "if customers differ by one unit of this, how much do they differ in spend?" That question asks for a rate of exchange between variables, denominated in their native units — dollars per order, dollars per day of recency — and the instrument that estimates it is linear regression. The name and the idea are old: Galton (1886), studying heredity, noticed that extreme parents had less extreme children — values "regressing" toward the mean — and the line he drew through that cloud of points became the general tool for summarizing how one variable runs with another. This guide develops it as working machinery, one step above intuition and well short of formalism, per the program's no-prerequisite design.

Simple linear regression models one outcome variable Y — the response — as a straight-line function of one predictor X, plus an error term that absorbs everything the line does not capture. The fitted model is an equation of the form $Y = b_0 + b_1X$, and its two coefficients are the deliverable. The intercept b_0 is the model's estimate of Y when X is zero — sometimes meaningful (predicted spend at zero discount exposure), sometimes a mathematical anchor outside the data's range (predicted spend at zero days of tenure describes a customer who signed up this morning), and the analyst states which. The slope b_1 is the working number: the estimated difference in Y associated with a one-unit difference in X. Its units are always outcome-units per predictor-unit, and this guide's standing rule is that a coefficient is not read until its units are said aloud. A slope of −0.85 in a regression of monetary value on recency_days is not "negative 0.85"; it is "eighty-five cents less total spend per additional day since last purchase, across the observed range" — a sentence with dollars, a direction, a denominator, and a scope.

DEFINITION

Simple Linear Regression

Simple linear regression models a numeric response variable as a linear function of one predictor: an intercept (the estimated response when the predictor is zero) plus a slope (the estimated difference in response per one-unit difference in the predictor), with an error term absorbing what the line does not capture. Coefficients are estimated from data and read in the native units of the variables.

Source: Adapted from James et al. (2021).

Where does the line come from? Any line through the scatter leaves gaps between itself and the actual customers, and each gap has a name: the residual, the customer's actual Y minus the value the line predicts for their X. Positive residual, the model underestimated them; negative, it overestimated. The least squares criterion chooses the one line that makes the sum of squared residuals as small as possible — squared for the same reason inertia squared distances in Section 6.8: large misses are penalized disproportionately, and the arithmetic is computable. Uniqueness carries a condition worth attaching now, because Sections 7.9 and 7.10 will meet the exception. When the model's predictor matrix has full column rank — no predictor is an exact linear combination of the others — least squares yields a unique coefficient vector. When predictors are perfectly redundant, as in the dummy-variable trap, software may still return a fitted result by numerical convention, but the individual coefficients are not uniquely identified and should not be interpreted. The intuition worth keeping is the balance-point image: the least squares line is the scatter's center of gravity stretched into a slope, it passes through the point of means, and — like every mean since Chapter 5 — it is sensitive to extreme points, which is Anscombe's third panel and the whales again. The formalism beyond this is machinery the software owns; what the analyst owns is the criterion and its sensitivities.

DEFINITION

Residual

A residual is the difference between an observation's actual response value and the value the fitted model predicts for it: actual minus predicted. Residuals are the model's itemized account of what it missed, and their patterns — trends, curves, funnels, clusters — are diagnostic evidence about where the model's form is wrong.

Source: Adapted from James et al. (2021).

Residuals deserve the second definition box because they are this chapter's diagnostic instrument, the regression counterpart of Chapter 4's verification log. Software can almost always return a fitted result; whether its coefficients are uniquely identified and substantively meaningful depends on the model's design, and whether the line is an honest summary shows up in what it leaves behind. The residual plot — residuals against predicted values, or against the predictor — should look like weather: patternless scatter around zero. A curve in the residuals means the relationship was not a line (the model form is wrong, not the world); a funnel that widens with predicted spend means the model misses big spenders by more than small ones, a near-universal feature of revenue data that Chapter 5's skew already predicted, and one with a direct consequence for the uncertainty columns that Section 7.10 will take up; a cluster of large residuals that turns out to be one segment means the single line is averaging over populations that want different lines — the mixed-population lesson again, now visible as diagnostic evidence and fixable with Section 7.9's machinery. Reading residual plots is a chart-reading skill in exactly Chapter 4's sense — the objects were introduced there; this chapter supplies what to look for when the chart is about a model.

Regression and correlation are close arithmetic relatives — for one predictor, the slope is r rescaled by the ratio of standard deviations, so they always agree in sign — but they answer different questions and fail differently. r is unit-free and symmetric: it grades the tightness of the cloud. The slope is unit-bearing and directional in form: it prices the relationship in decision units. That pricing is why the meeting quotes slopes, and why the analyst's caution transfers whole: the regression of Y on X and the regression of X on Y are different lines answering different questions, and nothing in either line knows which variable is in charge — the model's direction is the analyst's framing, not a discovery. Every rival explanation from Section 7.2 survives the fitting of a line untouched.

7.7 How Good Is the Line? R^2, Fit, and Usefulness

A fitted line and honest units are not yet a report card. The standard single-number grade is R^2, the coefficient of determination: the share of the response variable's variation that the model accounts for. The construction is worth seeing once in words. Total variation is the sum of squared deviations of Y around its own mean — how much there is to explain, the baseline ignorance of a model that predicts the mean for everyone. Residual variation is the sum of squared residuals — the ignorance that remains after the model. R^2 is the fraction of the original variation the model removed: R^2 = 1 minus (residual variation ÷ total variation).

Three properties are usually stated as though they were laws of arithmetic, and they are better stated with their conditions attached, because students meet software that reports R^2 under other conventions. For in-sample ordinary least squares models fitted with an intercept to the same observations, R^2 lies between 0 and 1 and cannot decrease when predictors are added to a nested model; and in a simple one-predictor model with an intercept, R^2 equals r^2. Remove the intercept and the arithmetic changes: statsmodels, for example, computes a centered R^2 when a constant is present and an uncentered version when it is absent, so the two are not comparable and the uncentered figure is routinely and misleadingly larger (Seabold & Perktold, 2010). Change the observations, or evaluate the model on data it was not fitted to, and the guarantee that R^2 stays inside the unit interval disappears with it — a fact Chapter 8 will need immediately. The conditions are not fussiness. They are the difference between a number that means what the sentence says and a number that merely looks familiar.

DEFINITION

R^2 (Coefficient of Determination)

R^2 is the proportion of a response variable's total variation, measured around its mean, that a fitted model accounts for: one minus the ratio of residual variation to total variation. It compares the model to the baseline of predicting the mean for every observation. For in-sample ordinary least squares models fitted with an intercept to the same observations, R^2 lies between 0 and 1 and cannot decrease when predictors are added; in a simple one-predictor model with an intercept, it equals r^2. Because it cannot decrease, a rising R^2 is not, by itself, evidence of a better model.

Source: Adapted from Wooldridge (2020) and James et al. (2021); software conventions per Seabold and Perktold (2010).

Notice what R^2 secretly is: a baseline comparison, in exactly Section 2.7's sense. "The model accounts for 34 percent of spend variation" means "against the interpretive baseline of guessing the mean, residual error shrank by a third." This guide flags the connection because baselines are a running thread — Chapter 8 will operationalize them for prediction and Chapter 10 will extend them to time — and because remembering what R^2 is a comparison against inoculates against its two standing misreadings. The first misreading is that high R^2 certifies a good model. It does not: R^2 measures fit to this data, and Anscombe's quartet holds four identical R^2 values over four different truths; a curve, an outlier, or a leaking variable can each buy fit without buying validity. In customer-level data the suspicion runs mostly the other way — an R^2 above roughly 0.9 on individual behavior should trigger the Table 7.1 reflex, because individual humans are not that predictable, and near-perfect fit usually means the model was handed the answer, the trap Code 7.18 demonstrates on purpose.

The second misreading is that low R^2 certifies a useless model. It does not: R^2 grades total variance accounted for, while decisions often turn on the slope. A model accounting for 12 percent of spend variation can still estimate the recency slope precisely enough to price a win-back program — noisy individuals, reliable rates. In other words: fit is a property of the model against this data; usefulness is a property of the model against a decision, and the two judgments can disagree in both directions.

CONCEPT

Fit Is Not Usefulness

R^2 answers one narrow question — how much of the outcome's variation this model absorbs in this sample. Decisions rarely ask that question. They ask whether a specific coefficient is estimated well enough to act on, whether the model's form survives its residuals, and whether the relationship is stable enough to hold next quarter. A high-R^2 model can fail all three; a low-R^2 model can pass all three. Report R^2 — it is the honest disclosure of how much is not accounted for — but never let it stand alone as the verdict, and treat "too good" with the same suspicion as "too poor." The number that flatters the model is the number to audit first.

Source: Course concept developed for this guide, informed by Shmueli (2010).

7.8 Multiple Regression and the Ceteris Paribus Promise

The opening case's claims all founder on the same rock: each compares customers who differ in many things at once and attributes the difference to one of them. The instrument built for exactly this problem is multiple regression: one response, several predictors, fitted jointly. The model form extends naturally — $Y = b_0 + b_1X_1 + b_2X_2 + \dots$ — and least squares still chooses the coefficients that minimize squared residuals. What changes is the meaning of each coefficient, and the change is the entire reason the method matters: in a multiple regression, b_1 estimates the

difference in Y per one-unit difference in X_1 among observations that share the same values of every other predictor in the model. The Latin tag is ceteris paribus — other things equal — and the operation is statistical control (Wooldridge, 2020).

One word has to be handled carefully here, because the natural English for statistical control is misleading. Textbook prose and classroom shorthand both reach for "matched" or "comparable customers," and ordinary least squares does not match anybody. It fits a surface to the whole cloud and reports the surface's slope in one direction. The estimate is therefore a model-based comparison — an adjusted difference computed under the model's assumed functional form — and it can be produced even where few or no real customers share the relevant combination of covariate values. Where that happens, the model is extrapolating across a region the data do not populate rather than comparing genuinely similar customers, and nothing in the printed coefficient distinguishes the two cases. The honest phrasing, used throughout this chapter, is that a coefficient compares customers statistically, holding the included predictors constant within the model's assumed form.

DEFINITION

Multiple Regression

Multiple regression models a numeric response as a linear function of two or more predictors fitted jointly, so that each coefficient estimates the difference in the response per one-unit difference in its predictor, holding the other included predictors constant within the model's assumed functional form. It implements statistical control by adjustment rather than by matching: the comparison is model-based, extends only to the variables the model was given, and may extrapolate where the data contain few observations sharing the relevant combination of values.

Source: Adapted from Wooldridge (2020).

Applied to the app claim, the machinery earns its keep immediately. The raw comparison — app users versus non-users, a 2.4× spend ratio — mixes the app difference with every difference that travels with it: app users are younger, more urban, more loyalty-enrolled, longer-engaged, more email-active. Put spend on the left and app ownership on the right alongside recency, tenure, email opt-in, and segment membership, and the app coefficient now estimates the spend difference between app owners and non-owners adjusted for all of those. On StyleCraft's data the labs will find that this adjusted difference is far smaller than the raw ratio implied — most of "2.4×" was the Urban Loyal Core's profile traveling with the app, not the app itself. That shrinkage from raw gap to adjusted gap is the most useful single exhibit a drivers analysis produces, and the meeting-ready sentence practically writes itself: "customers with the app spend more; most of that gap reflects who installs the app rather than anything the model can attribute to installing it."

The promise, however, has strict boundaries, and the guide states them as sharply as the promise itself. Ceteris paribus means other included things equal. The model holds constant what it was given and nothing else; every relevant variable not in the model remains free to vary between the compared customers and to smuggle its influence into the coefficients. This failure has a name — omitted variable bias — and a usable direction logic, provided the logic is stated with its scope. In the simplest linear case, one included predictor and one omitted variable, the direction of the bias follows the product of two relationships: the omitted variable's effect on the response, times its association with the included predictor. In a multiple regression the relevant association is narrower: it is the part of the omitted variable's association with the included predictor that remains after the model's other predictors are taken into account. That qualification matters in practice, because a variable that looks strongly related to the predictor in a raw cross-tabulation can be nearly unrelated to it once the model's other controls are in place, and the bias reasoning must follow the adjusted relationship rather than the raw one.

DEFINITION

Omitted Variable Bias

Omitted variable bias is the distortion of a fitted coefficient that arises when a variable related to both the response and an included predictor is left out of the model: the included predictor absorbs credit or blame belonging to the omitted one. In the simplest linear case, the direction of the bias follows the product of the omitted variable's effect on the response and its association with the included predictor. In a multiple regression, the relevant association is the part that remains after the model's other predictors are taken into account.

Source: Adapted from Wooldridge (2020).

If underlying brand engagement increases spend and is positively associated with email activity, then omitting it would tend to bias the email coefficient upward: under those assumptions the fitted model attributes to email part of an association that belongs to enthusiasm. Note the shape of that sentence. The direction follows from stated assumptions about the omitted variable, not from a causal fact the data have established — which is exactly what makes it usable, since the reasoning can be run before the missing variable is ever measured. The direction logic converts anxiety into analysis: for every coefficient the meeting will quote, the analyst lists the plausible omitted variables, reasons the sign of each bias, and reports which way the estimate leans. Statistical control is not a causal machine; it is a better class of comparison, and the honest sentence names both halves.

CONCEPT

The Coefficient Is an Adjusted Comparison, Not a Lever

A multiple regression coefficient describes a model-adjusted difference between customers, computed as though the model's other variables were held constant. It does not describe what happens when you move the variable, for three reasons that never go away: the adjustment extends only to the model's own variable list; the adjustment is a property of the fitted surface rather than of matched pairs of real customers, so it can be reported for combinations the data barely contain; and even a perfectly adjusted difference is still an association — the customers chose their own X. Read every coefficient as "customers who differ by one unit of X differ, on average, by b in Y, holding the listed variables constant" — and treat any restatement beginning "if we increase X" as a new claim requiring new evidence. This box is the chapter's verification theme applied to language, and Section 7.15 shows what its violation costs.

Source: Course concept developed for this guide, informed by Wooldridge (2020) and Freedman (1991).

Two practical notes complete the working level. First, model inputs are governed by measurement, exactly per Section 3.4: numeric predictors enter as they are; ordinal predictors (loyalty tier) enter either as ordered scores — asserting equal steps, an assertion to declare — or as categories per Section 7.9; nominal predictors (segment, metro, channel) must enter as dummy variables, because arithmetic on category codes is the level-of-measurement error Chapter 3 warned against. Second, adding predictors is not free: each addition changes every other coefficient's meaning ("holding constant" now includes the newcomer), R^2 rises mechanically whether or not the addition carries information, and predictors that share information destabilize one another — the multicollinearity of Section 7.10. The model is a specified instrument, not a bucket; every variable in it should be there because the analysis's question, written per Section 2.5's specification discipline, put it there.

7.9 Dummy Variables, Reference Categories, and Interaction Terms

The drivers question at StyleCraft is inseparable from the segments, and the segments are categories: four names, no natural ordering, no units. Regression's arithmetic needs numbers, and the encoding that supplies them honestly is the dummy variable: an indicator column equal to 1 when the observation belongs to a category and 0 otherwise. A categorical variable with c categories enters the model as c − 1 dummies; the category left without a dummy is the reference category, and it is not omitted from the analysis — it becomes the analysis's baseline. Every dummy coefficient is then read as an adjusted difference against the reference: with Urban Loyal Core as the reference, the Suburban Occasion dummy's coefficient is the estimated difference in the response between Suburban Occasion customers and Urban Loyal Core customers, holding the model's other variables constant. One category must always be left out — including all c dummies alongside an intercept gives the model two ways to say the same thing, a perfect redundancy that breaks the fitting (the dummy variable trap, multicollinearity in its terminal form).

What the intercept means in such a model needs stating precisely, because the convenient shorthand is wrong as soon as the model contains anything else. It is tempting to say that the reference category's average outcome is carried by the intercept. That is true only in the special case of a model containing nothing but that one set of dummies. Add a numeric predictor and the intercept becomes the model's expected response for the reference category when that numeric predictor equals zero — which may describe no customer who exists. Add a second categorical predictor and the intercept refers to the reference level of both. The reference group's observed mean and the model's intercept are different quantities, and confusing them is how an analyst ends up reporting a negative baseline spend with a straight face.

DEFINITION

Dummy Variable and Reference Category

A dummy variable encodes category membership as an indicator: 1 if the observation is in the category, 0 otherwise. A categorical predictor with c categories enters a regression as c − 1 dummies, and the reference category is represented when all dummies for that predictor equal zero. With an intercept in the model, each dummy coefficient is an adjusted difference from that reference category. The intercept represents the model's expected response for the reference category when all other predictors are at their coded zero or reference values — which is the reference group's observed mean only in the special case of a model containing no other predictors.

Source: Adapted from Wooldridge (2020).

The choice of reference category changes no substance — every model with any reference fits identically and tells the same story — but it changes every printed number and therefore every sentence, so it is chosen for readability and declared. The guide's convention: the reference is the largest or most decision-central category, so that coefficients read as "compared to our core." Table 7.3 shows the encoding for StyleCraft's four segments with Urban Loyal Core as reference, and the reading column is the deliverable skill: students who can compute dummies and cannot read them against the reference produce the most common wrong sentence in applied regression — "the model shows Suburban Occasion customers spend $X" instead of "$X more or less than the Core, adjusted for the model's controls."

Table 7.3

Dummy encoding of the four segments, with Urban Loyal Core as the reference category

Segment (from Chapter 6)	seg_ suburban	seg_ discount	seg_new	How the coefficient reads
Urban Loyal Core	0	0	0	Reference: no coefficient of its own. The model's expected response for this group appears only when every other predictor is also at zero or at its own reference level.
Suburban Occasion	1	0	0	Adjusted difference from the Urban Loyal Core, holding the model's other predictors constant
Discount-Dependent	0	1	0	Adjusted difference from the Urban Loyal Core, holding the model's other predictors constant
New / At-Risk	0	0	1	Adjusted difference from the Urban Loyal Core, holding the model's other predictors constant

Dummies let group membership shift the model's level. The next construction lets it shift the model's slopes — because the sharpest drivers questions are not "does this lever matter?" but "for whom does the relationship differ?" An interaction term is a new predictor built as the product of two others, and its coefficient estimates how the association between one variable and the response differs across values of the other.

Interactions have to be read conditionally, and the reading is mechanical enough to state as a rule. Suppose the model contains app use, loyalty enrollment, and their product, both coded 0 or 1. Then the app coefficient is the app–spend association among customers whose loyalty enrollment equals zero — the non-enrolled. The loyalty coefficient is the loyalty–spend association among customers whose app use equals zero. And the interaction coefficient is the difference between the app association among the enrolled and the app association among the non-enrolled: the app association for enrolled customers is the sum of the app coefficient and the interaction coefficient. Nothing in the model reports an "overall" app association once the interaction is present, and an analyst who quotes the app coefficient as though it were one has quoted a conditional number as an unconditional claim.

DEFINITION

Interaction Term

An interaction term is a predictor formed as the product of two others, whose coefficient estimates how the association between one predictor and the response differs across values of the other. A model with an interaction no longer has one slope per variable; it has slopes that depend on context. Each main-effect coefficient is the association for its variable when the interacting partner equals zero, and the association at any other value of the partner is obtained by adding the interaction coefficient the appropriate number of times.

Source: Adapted from Wooldridge (2020) and James et al. (2021).

The registry example is the one the budget meeting cares about, and this guide names it one way and holds to it: loyalty enrollment × app-use status, with app use serving as the selected channel-engagement indicator. The dataset's specification plants a loyalty-by-channel structure, and app use is the channel variable StyleCraft's schema actually carries at the customer grain, so the model estimates the planted structure through that indicator rather than through a channel construct the table does not contain. The question the interaction answers is whether the association between app engagement and spend differs by loyalty enrollment — and the honest report of the answer is two conditional associations in dollars, not one significance verdict.

Two standing cautions travel with every interaction. The main-effect coefficients change meaning the moment their interaction enters, which is the conditional reading just given, and it is the single most common source of misquotation in applied regression. And interactions multiply the model's appetite for data: estimating a separate

association per group requires enough customers in each combination of the two variables, a substantiality check Chapter 6's segment sizes already supplied and Code 7.15 re-checks by printing the cell counts before the model is fitted. Where an interaction is materially nonzero, the finding to write is descriptive and conditional — "the association between discount exposure and spend differs by loyalty status" — rather than the causal restatement that a slide will reach for, and the exercises test the difference.

7.10 Reading Regression Output as Evidence: p-Values, Confidence Intervals, and Multicollinearity

A fitted model arrives as a table: one row per coefficient, and columns the software prints whether or not anyone can read them. This section supplies the working-level reading of the columns that matter after the estimate itself — the uncertainty columns — and the two structural judgments that govern how much any of it means: whether the predictors undermined each other, and what job the model was built to do. Everything here is calibrated to the analyst's seat: enough to weigh evidence and defend a table in a meeting, on the explicit understanding that the full statistical machinery belongs to later coursework.

Every coefficient is an estimate from one sample of customers, and would come out somewhat differently on another sample — the fitted $41 per order is the center of a range of values the data cannot distinguish, not a constant of nature. The standard error measures that sampling wobble, and the two working instruments are built from it. The 95 percent confidence interval is the honest way to report a coefficient: the range of values compatible with the data at conventional confidence — "$34 to $48 per order" — whose width is the finding about precision, and whose reading follows Chapter 5's spread discipline: a wide interval around a large estimate says "big but poorly pinned down," and a narrow interval around a small one says "small and we know it."

Before either instrument can be trusted, one assumption behind the printed standard errors has to be checked, and Section 7.6 already predicted its failure. Ordinary least squares computes standard errors, by default, on the assumption that the residual variance is the same across the range of fitted values. Revenue data rarely obliges: the residual plot funnels, because the model misses big spenders by more than small ones. That is heteroskedasticity, and its consequence is specific — it does not bias the fitted coefficients, and it does distort the standard errors, so the intervals and p-values built from them are wrong in a direction the default output does not disclose. The repair is a different covariance estimator, requested at fit time. Statsmodels defaults to a nonrobust estimator and supports several robust alternatives; this guide uses HC3, which performs well in samples of the size marketing analysts actually work with (MacKinnon & White, 1985; Seabold & Perktold, 2010). A chapter that teaches students to expect a residual funnel and then quotes nonrobust intervals two pages later would be teaching the diagnosis and skipping the treatment, so every model in Lab 7.2 is fitted with robust standard errors and Code 7.9 states plainly what changes and what does not: the coefficients and the predictions are identical, and the standard errors, confidence intervals, and p-values are not.

The coefficient's p-value answers a narrower question than either instrument above, and it is worth stating exactly, because the loose version students carry — "the probability of getting an estimate this far from zero by chance" — omits the machinery that produces it. The software computes a test statistic from the coefficient and its estimated standard error, and refers that statistic to a distribution implied by the null hypothesis and the model's assumptions. Small p-values say the data are hard to reconcile with "no association, given this model"; they say nothing about size, importance, or cause. The American Statistical Association's consensus statement, issued after decades of misuse, compresses to three warnings this guide adopts: a p-value is not the probability that a hypothesis is true, statistical significance is not practical significance, and no single p-value should by itself determine a scientific, business, or policy conclusion (Wasserstein & Lazar, 2016). On eight thousand customers, near-trivial associations clear 0.05 easily — at this sample size, significance is table stakes, and the analyst's questions are the interval's location and width in decision units. The asterisk column is where reading starts, not where it ends.

DEFINITION

Coefficient p-Value and Confidence Interval

A p-value is the probability, assuming the stated null hypothesis and statistical model are correct, of obtaining a test statistic at least as extreme as the one observed. For a regression coefficient the null hypothesis is that the coefficient is zero given the model's other variables, and the test statistic is computed from the estimate and its standard error — so the p-value inherits every assumption behind that standard error, including the assumption about residual variance that robust estimators relax. A 95 percent confidence interval is the range of coefficient values compatible with the data at conventional confidence; its center is the estimate and its width is the precision. Together they grade the strength of evidence for a relationship, never the relationship's causal status.

Source: Adapted from Wasserstein and Lazar (2016) and Wooldridge (2020).

The first structural judgment is multicollinearity: predictors that carry overlapping information. When two variables move together tightly — frequency and monetary; email opt-in and app use; tenure and recency in a base that grew in waves — the model struggles to apportion the shared movement between them, and the symptoms are characteristic: coefficients with large standard errors and unstable values that swing when a variable enters or leaves; individually "insignificant" predictors in a model whose joint fit is strong; occasionally a sign that flips against all descriptive evidence.

The predictor correlation matrix is the first diagnostic, read with Table 7.1's "suspect" row, and it is not sufficient by itself. Pairwise correlations detect only pairwise overlap, and the damaging case is often multi-variable: one predictor is closely approximated by a combination of several others while no single pairwise correlation looks extreme. The instrument that sees this is the variance inflation factor, which measures how much a coefficient's estimated variance is inflated by the predictor's relationships with all the other predictors together (Seabold & Perktold, 2010). This guide declines to publish a universal cutoff, because none survives contact with the literature. VIF is a diagnostic rather than a verdict: large values indicate that the model has difficulty separating a predictor's coefficient from the information carried by the others, and what counts as large depends on how precisely the coefficient needs to be estimated for the decision at hand. The paired instrument is the instability test — refit with and without the twin and watch the survivor's coefficient — which Code 7.17 runs as a specification-sensitivity check.

DEFINITION

Multicollinearity

Multicollinearity is strong overlap in the information carried by a model's predictors — whether between a pair or between one predictor and a combination of several others — which leaves their joint explanatory contribution largely intact but makes the division of that contribution among individual coefficients unstable and imprecise: inflated standard errors, estimate swings across specifications, and occasionally reversed signs. It is diagnosed from predictor correlations, variance inflation factors, and specification-sensitivity checks, and addressed by combining, dropping, or jointly reporting the entangled predictors. Multicollinearity may leave in-sample fit largely intact while destabilizing coefficients; it can also weaken out-of-sample prediction when the relationships among predictors shift.

Source: Adapted from Wooldridge (2020); variance inflation factor per Seabold and Perktold (2010).

The response to a multicollinearity finding is a decision, not a formula: combine the twins into one variable, drop one and say so, or keep both and report the pair's joint story while refusing to interpret their individual split. And one special case is a defect rather than a judgment call: a predictor that is arithmetically derived from the

response produces a near-perfect fit that measures the derivation, not behavior. Regressing monetary value on the product of average order value and frequency is the pure form of the error, because that product is monetary by construction (Table 6.5), and Code 7.18 demonstrates it deliberately so that students meet the sensation of a model that is too good. Chapter 8 will name the general disease leakage; here it is caught by the derivation-tracing habit Table 7.2 installed.

The second structural judgment sorts everything this chapter has built, and it is the distinction the rest of Part II stands on: prediction and explanation are different modeling goals, and the same equation serves them differently (Shmueli, 2010). A predictive model has one obligation — accurate outputs on customers it has not seen — and it tolerates much that an explanatory model cannot: entangled predictors, uninterpretable coefficients, variables that merely correlate with the future. An explanatory model has a different obligation — coefficients an analyst can defend as honest structure — and it tolerates lost accuracy in exchange: variables chosen by the specification's question, collinear twins separated or set aside, every retained coefficient readable in units with its bias direction reasoned. The drivers analysis of this chapter is explanatory work; the spend model of Chapter 8 will be predictive work; and most applied failures of regression trace to answering one goal's question with the other goal's model — quoting individual coefficients out of a kitchen-sink predictive fit, or crippling a forecasting model to keep its coefficients tidy. Table 7.4 sets the comparison, and the lab's final verification check asks which column the opening case's deliverable lives in.

Table 7.4

Prediction and explanation as distinct modeling goals

Dimension	Explanation (this chapter)	Prediction (Chapter 8)
Question	How do these variables relate to the outcome?	What will the outcome be for new cases?
Success measure	Defensible coefficients: signs, units, bias reasoning	Out-of-sample accuracy against a baseline
Variable selection	By the specification's question	By predictive contribution
Multicollinearity	A serious problem — it muddles attribution among coefficients	Often tolerable in-sample, but a risk when predictor relationships shift out of sample
Coefficient reading	The deliverable	Optional; often declined
Characteristic failure	Causal overclaiming (Section 7.15)	Leakage and overfitting (Chapter 8)

7.11 AI as a Regression Assistant

Chapter 6 marked the point where AI assistance began delivering whole analytic acts; this chapter is where the acts start arriving in the language of decisions. Prompted with the feature table and "what drives spend?", a current assistant will select variables, fit models, format the coefficient table, and — this is the part that changes the analyst's job — narrate the results in fluent managerial prose, complete with recommendations. The narration is the hazard. Code errors announce themselves; a regression that runs, fits, and is narrated with confident causal language fails silently, and it fails in the exact register the meeting will quote. The division of labor therefore follows the pattern set in Section 6.12, with this chapter's twist: the assistant may draft the machinery, and the analyst audits not only the numbers but the verbs.

Where the assistant genuinely helps, use it. Drafting the modeling code — formula syntax, dummy encoding, the residual-plot boilerplate, the tidy coefficient table with intervals — is legitimate delegation of mechanics this chapter has specified, provided the variable list comes from your written specification rather than the assistant's initiative. Generating the audit exhibits themselves — predicted-sign checklists, with-and-without-whales correlation runs, specification-sensitivity refits — is exactly the tedious-but-mechanical work assistants accelerate. Translating a finished, verified coefficient table into plain-language draft sentences is useful, provided every sentence is then checked against the verb discipline of Section 7.2. And the assistant is a genuinely good

brainstorming partner for candidate confounders — "what variables might influence both discount exposure and total spend?" — because enumerating rivals is a breadth task, and breadth is what it has.

What must never be delegated are the judgments, and the failure modes are characteristic enough to name. The confident narration: asked for drivers, assistants return causes — "app adoption drives a $212 spend lift" — because fluent causal prose saturates their training data; the arithmetic may be flawless while the verb is unearned. The silent specification: the assistant chose which variables entered, which were dropped, what the reference category was, and whether an interaction was included — choices that change every number — and reported none of them as choices. The plausible-but-impossible coefficient: a sign or magnitude that contradicts certified facts (a positive recency coefficient; an intercept implying negative spend; a segment gap triple anything Chapter 5 measured), delivered with the same serene formatting as a correct one. The units slip: narrating a coefficient without units, or with the wrong ones — "recency matters most because its coefficient is largest" when recency is in days and frequency is in orders, the standardization confusion of Section 6.7 reborn in a coefficient table. The default covariance estimator: a model fitted on visibly funneling residuals and reported with nonrobust intervals, because nothing in the default workflow connects the diagnostic to the inference. The diagnostics skip: no residual plot, no mention of the funnel or the curve, because the default workflow fits and reports but does not examine. The unconditional reading of a conditional coefficient: a main effect quoted as an overall association while an interaction sits in the same model. And the significance theater: p-values below 0.05 presented as importance, exactly the misreading this section retired.

The audit routine that answers them is this chapter's verification theme, compact enough to run on every AI-fitted model and every colleague's model — four points, in order, before any narration is accepted. Table 7.5 states it as a working checklist; the AI in Practice box operationalizes it as a prompt-and-audit pattern.

Table 7.5

The four-point audit of a fitted regression

Point	The check	Fails when
1. Signs	Every coefficient's direction against written predictions and certified descriptives (Chapter 5)	A sign contradicts known structure with no confounding story offered
2. Magnitudes	Every coefficient's size against hand-computable anchors: group means, Chapter 5 gaps, plausible ranges	A coefficient implies differences no descriptive table shows, in either direction, with no explanation grounded in the model's variables
3. Units	Every coefficient restated aloud in outcome-units per predictor-unit; comparisons only on common scales; conditional coefficients read conditionally	A narration compares raw coefficients across different units, or quotes a main effect from a model containing its interaction as though it were unconditional
4. Diagnostics	Residual plots inspected; the covariance estimator matched to what the residuals show; variance inflation factors read; R^2 read against Section 7.7; specification sensitivity spot-checked	Fit is reported but residuals were never looked at, or intervals are quoted from a nonrobust fit on funneling residuals

AI IN PRACTICE

The Regression That Must Survive Its Audit

A two-prompt pattern with the audit between. Prompt one, the specification: paste the variable list with derivation rules (Table 6.5), the declared response, the reference category, the covariance estimator, and the exact model formula from your written spec, then ask for mechanics only — "Fit this model as specified with HC3 standard errors; report the coefficient table with 95 percent intervals in native units, the residual-versus-fitted plot, variance inflation factors, and R^2; change nothing in the specification; narrate nothing yet."

> Then run the four-point audit of Table 7.5 yourself: check every sign against the predictions you wrote before fitting; check magnitudes against Chapter 5's certified gaps; say every coefficient in its units, and read any coefficient whose interaction is in the model as the conditional quantity it is; read the residual plot and confirm the covariance estimator matches what it shows.
>
> Prompt two, only after the model survives: "Draft three sentences reporting these estimates as associations, using 'is associated with' or 'predicts,' naming units and the held-constant list, and making no causal claim." Audit the verbs in the reply as strictly as the numbers — assistants reintroduce causal language at the narration step even when the numbers are clean. Predictions first, per Section 1.7; every exchange documented per Appendix D; and the analyst of record signs the sentences, not the software that drafted them.
>
> Source: Course concept developed for this guide, informed by Wasserstein and Lazar (2016) and Shmueli (2010).

The deeper continuity: in Chapter 6 the assistant proposed structure and the analyst tested whether the structure was real; here the assistant proposes relationships and the analyst tests whether the relationships are honestly stated. The labs that follow are built so the audit has teeth. The dataset's specification planted the coefficient structure the labs recover — which variables carry spend, at what strengths, with one designed interaction — so an AI-fitted model's claims can be graded against a known answer, and Exercise 7.6 does exactly that with a deliberately underspecified prompt.

7.12 Hands-On Application in Python and Google Colab

The preceding sections built the instruments; this section points them at the opening case, in two labs that mirror the chapter's two halves. Lab 7.1 disciplines the raw material: correlations on the feature table, predicted before computed, plotted before believed, and stress-tested against whales and mixed populations. Lab 7.2 builds the deliverable: the drivers models for spend and average order value, with segment dummies, robust inference, the designed interaction, and the four-point audit run against an AI-fitted version. Both labs consume the customer_features table built and reconciled in Lab 6.1 — the analysis date remains June 30, 2026, the window remains the full 24 months — plus, in Lab 7.2, two roll-ups from the certified transactions file. AI assistants may draft any code cell (Appendix C has templates; Appendix A covers Colab mechanics); every output is predicted before it is computed, and every exchange is documented per Appendix D.

Two conventions hold across every cell. Models are fitted with heteroskedasticity-robust standard errors, per Section 7.10, because this chapter predicts a residual funnel before it fits anything. And the four segments enter as explicitly constructed indicator columns with short names rather than through a formula shortcut, so that every coefficient label in the printed output is one a student can read aloud — the formula expression C(segment, Treatment(reference="Urban Loyal Core")) produces exactly the same fit with longer labels, and either is correct.

7.12.1 Lab 7.1, Part A: Correlation by Hand in Miniature

The miniature is Lab 6.1's ten-customer feature table, reused so every number is hand-checkable. The question: how strongly do frequency and monetary value move together? Write your prediction first — most students, reasoning "more orders means more revenue," predict r above 0.7.

Code 7.1. Covariance and correlation on the miniature feature table

```
import numpy as np
import pandas as pd

mini = pd.DataFrame({
    "customer_id": ["C001", "C002", "C003", "C004", "C005",
                    "C006", "C007", "C008", "C009", "C010"],
```

```
    "frequency": [4, 2, 3, 1, 3, 1, 1, 2, 1, 2],
    "monetary":  [196.00, 400.00, 69.60, 44.00, 127.00,
                  242.00, 24.00, 126.00, 38.00, 312.00],
}).set_index("customer_id")

# The two covariances of Section 7.3, computed side by side so that the
# denominator is a visible choice rather than a silent default.
dev_f = mini["frequency"] - mini["frequency"].mean()
dev_m = mini["monetary"] - mini["monetary"].mean()
deviation_products = dev_f * dev_m
n = len(mini)

print(mini.mean().round(2))
print("sum of the ten deviation products:",
      round(deviation_products.sum(), 2))
print("sample covariance    (n - 1):",
      round(deviation_products.sum() / (n - 1), 2))
print("population covariance    (n):",
      round(deviation_products.sum() / n, 2))
print("pandas .cov():", round(mini["frequency"].cov(mini["monetary"]), 2))
print("pandas .corr():", round(mini["frequency"].corr(mini["monetary"]), 3))

assert np.isclose(mini["frequency"].mean(), 2.0)
assert np.isclose(mini["monetary"].mean(), 157.86)
assert np.isclose(deviation_products.sum(), 240.60, atol=0.01)
# pandas reports the SAMPLE covariance, which is what Section 7.3 declared.
assert np.isclose(mini["frequency"].cov(mini["monetary"]),
                  deviation_products.sum() / (n - 1))
```

Expected output: means of 2.00 orders and 157.86 dollars; a deviation-product sum of 240.60; a sample covariance of 26.73 against a population covariance of 24.06; pandas reporting 26.73; a correlation of 0.199; and four assertions passing silently.

Input: ten literal customers. Transformation: paired deviations, their products, and two divisions. Output: the two covariances and the correlation. The cell exists to make one distinction concrete before it can cause trouble at scale: the two covariances differ by roughly 11 percent on ten rows, and the correlation is indifferent to the choice because the same denominator appears in the covariance and in both standard deviations and cancels. A student who computes the population covariance by hand and then compares it with the pandas figure has found a discrepancy that is a convention rather than an error — which is exactly the discrimination this guide's verification habit is meant to develop.

VERIFICATION CHECK

Before you run: compute r by hand from the printed table. The means are frequency 2.0 and monetary 157.86. The ten products of paired deviations sum to 240.60, so the sample covariance is 240.60 ÷ 9 = 26.73. The sample standard deviations are 1.054 orders and 127.34 dollars. Therefore r = 26.73 ÷ (1.054 × 127.34) ≈ 0.199. Write down each intermediate figure, and predict which of the two printed covariances pandas will match.

After you run: every figure reconciles to the cent, pandas matches the sample covariance rather than the population one, and all four assertions pass. Then confront the surprise: r ≈ 0.20 — weak, by Table 7.1 — against the strong association almost everyone predicts. Two observations resolve it, and both belong in the notebook. First, C002 and C010 — the two big-basket occasion shoppers — sit exactly at the mean on frequency, so they contribute zero to the covariance while inflating monetary's standard deviation, diluting r. Second, C006, the lapsed one-time $242 basket, pulls in the negative direction. The miniature contains two customer worlds, and the pooled r averages across them.

> Investigate if: your hand-computed covariance disagrees with the printed one by roughly a ninth. That is the n versus n − 1 convention, not an arithmetic error, and the fix is to state which quantity you meant. A disagreement of any other size is arithmetic, and the deviation-product column is where to look.

Code 7.2. The labeled miniature scatterplot

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6.5, 4.5))
ax.scatter(mini["frequency"], mini["monetary"], s=45)
for cid, row in mini.iterrows():
    ax.annotate(cid, (row["frequency"], row["monetary"]),
                xytext=(5, 4), textcoords="offset points", fontsize=8)

# The mean lines turn the plot into the covariance calculation: points in
# the upper-right and lower-left quadrants contribute positive products.
ax.axvline(mini["frequency"].mean(), linewidth=0.8, linestyle="--")
ax.axhline(mini["monetary"].mean(), linewidth=0.8, linestyle="--")
ax.set_xlabel("frequency (orders in the window)")
ax.set_ylabel("monetary (total revenue, $)")
ax.set_title("Ten customers, two worlds")
plt.tight_layout()
plt.show()
```

Expected output: a ten-point scatter with every customer labeled and two dashed mean lines dividing it into quadrants; C002 and C010 sit on the vertical mean line, and C006 sits alone in the upper-left quadrant.

Input: the miniature. Transformation: none — this is a rendering. Output: the picture that makes the arithmetic obvious. The mean lines are the point of the cell rather than decoration: they partition the plot into the four quadrants of the covariance calculation, so a customer's contribution can be read off its position. Points on a mean line contribute nothing, which is why the two largest spenders in the table move r not at all in the positive direction while widening the vertical spread that r divides by. This is the entire argument of Section 7.4, experienced in the order statistics always should be: number, surprise, plot, understanding.

7.12.2 Lab 7.1, Part B: The Matrix at Full Scale, Stress-Tested

Part B moves to the full customer_features table. Before computing anything, write the predicted sign of the correlation between monetary and each of: recency_days, frequency, aov, discount_share, store_share, orders_per_month, revenue_per_month, app_user, and email_opt_in — with one sentence of reasoning each, drawing on Chapter 5's certified findings and Chapter 6's segment profiles. The two engagement flags belong in that list and in the matrix: the opening case's app and email claims are claims about exactly those columns, and a matrix that examines behavior while omitting the engagement indicators cannot speak to two of the three levers.

Code 7.3. Completeness and type checks before the matrix

```
corr_features = [
    "monetary",
    "frequency",
    "recency_days",
    "aov",
    "discount_share",
    "store_share",
    "orders_per_month",
    "revenue_per_month",
    "app_user",
    "email_opt_in",
```

```
]

# .corr() drops incomplete pairs silently and column by column, so a matrix
# built on partly missing columns quietly reports each cell on a different
# population. Check first; declare a policy if anything is missing.
missing = cf_full[corr_features].isna().sum()
print(missing[missing > 0] if missing.any() else "no missing values")

# Convert the Boolean flags deliberately. Left as Booleans they still
# correlate, but statsmodels will later label them app_user[T.True],
# which will not match the coefficient names this chapter uses.
cf_full[["app_user", "email_opt_in"]] = (
    cf_full[["app_user", "email_opt_in"]].astype(int))

assert cf_full[corr_features].notna().all().all(), (
    "declare a missing-data policy before calling .corr()")
assert set(cf_full["app_user"].unique()) <= {0, 1}
assert set(cf_full["email_opt_in"].unique()) <= {0, 1}

print(cf_full[corr_features].dtypes)
```

Expected output: the message "no missing values", then ten numeric dtypes with app_user and email_opt_in reported as integers, and three assertions passing.

Input: the certified feature table. Transformation: a completeness audit and one deliberate type conversion. Output: a feature list safe to correlate. The missingness check is not ceremony. Pandas computes each cell of a correlation matrix on the rows where that particular pair is complete — the documented behavior of DataFrame.corr, not an accident of this build (pandas development team, 2026) — so a single partly missing column produces a matrix whose cells describe different populations while presenting them as one exhibit, a grain violation in Section 3.3's sense hiding inside a routine call. The assertion above is the response: declare and resolve a missing-data policy before the call, rather than discovering afterward that row 4 of the exhibit was computed on a different customer base than row 7. The type conversion is the second half of the same discipline: a Boolean column correlates perfectly well, and the formula API in Lab 7.2 will rename it in a way that breaks every coefficient sentence this chapter teaches, so the conversion happens once, here, where it can be seen.

Code 7.4. The predicted-then-computed correlation matrix

```
corr = cf_full[corr_features].corr().round(2)

print("correlations with monetary, weakest to strongest:")
print(corr["monetary"].drop("monetary").sort_values())
print()
print(corr)

# Two structural properties, asserted rather than assumed: a correlation
# matrix is symmetric and its diagonal is exactly 1.
assert np.allclose(np.diag(corr.values), 1.0)
assert np.allclose(corr.values, corr.values.T)
```

Expected output: a sorted column of nine correlations with monetary, then a ten-by-ten symmetric matrix with ones on the diagonal, and two assertions passing.

Input: the ten checked features. Transformation: pairwise sample correlations. Output: the screening exhibit of Section 7.4. Grade your predictions against the sorted column first, one by one, before reading the full matrix — the sorted column is the drivers question in its most naive form, and seeing it before the corrections land is what makes the corrections stick. The exact values await the certified data, but the designed structure fixes the pattern your predictions should anticipate: recency negatively associated with monetary, frequency and the two per-month rates positively, and aov's correlation with frequency negative across the pooled base, because the base's variation is dominated by the contrast between frequent small-basket urban customers and infrequent large-basket suburban

ones. That negative aov–frequency cell is the segments speaking through a correlation matrix, and it belongs in your notebook's notes as the chapter's first mixed-population exhibit at full scale.

Two cells deserve immediate suspicion by Table 7.1's standards rather than celebration. The correlation between monetary and revenue_per_month will be high because one is the other divided by exposure, and the correlation between monetary and aov will be substantial because aov is monetary divided by frequency. Neither is a finding about customers; both are arithmetic relatives, and the derivation-tracing habit is what tells them apart from a behavioral result.

Code 7.5. Whale sensitivity: correlations with and without the top spend decile

```
cut = cf_full["monetary"].quantile(0.90)
trimmed_base = cf_full[cf_full["monetary"] < cut]

pooled = cf_full[corr_features].corr()["monetary"].drop("monetary")
trimmed = trimmed_base[corr_features].corr()["monetary"].drop("monetary")

compare = pd.DataFrame({
    "all customers": pooled.round(3),
    "top decile removed": trimmed.round(3),
})
compare["change"] = (compare["top decile removed"]
                     - compare["all customers"]).round(3)

removed = len(cf_full) - len(trimmed_base)
assert removed > 0
assert len(trimmed_base) + removed == len(cf_full)

print(f"top-decile cut at ${cut:,.2f}; {removed:,} customers removed "
      f"({removed / len(cf_full):.1%} of the base)")
print(compare.reindex(
    compare["change"].abs().sort_values(ascending=False).index))
```

Expected output: the cut point in dollars and the count removed, then nine rows sorted by the size of the change, with the largest movements on the spend-related features.

Input: the feature table and the concentration cut of Section 5.9. Transformation: the same correlation column computed twice, on the full base and with the top spend decile removed. Output: a movement report. Read the change column as an influence diagnostic, not as a correction: removing the whales does not produce the true correlation, because the whales are real customers and a substantial share of the revenue the chapter is trying to explain. What the column establishes is which stories depend on them. A correlation that survives the trim is a statement about the base; a correlation that halves or reverses is a statement about roughly one customer in ten, and it must be reported as such or not at all.

Code 7.6. Pooled and segment-faceted scatterplots of discount exposure and spend

```
segments = sorted(cf_full["segment"].unique())

fig, axes = plt.subplots(1, len(segments) + 1,
                         figsize=(3.6 * (len(segments) + 1), 3.4),
                         sharex=True, sharey=True)

pooled_r = cf_full["discount_share"].corr(cf_full["monetary"])
axes[0].scatter(cf_full["discount_share"], cf_full["monetary"],
                s=4, alpha=0.25)
axes[0].set_title(f"Pooled base\nr = {pooled_r:.2f}, n = {len(cf_full):,}")
axes[0].set_ylabel("monetary ($)")

for ax, seg in zip(axes[1:], segments):
    part = cf_full[cf_full["segment"] == seg]
```

```
    r = part["discount_share"].corr(part["monetary"])
    ax.scatter(part["discount_share"], part["monetary"], s=4, alpha=0.35)
    ax.set_title(f"{seg}\nr = {r:.2f}, n = {len(part):,}")

for ax in axes:
    ax.set_xlabel("discount_share")

plt.tight_layout()
plt.show()
```

Expected output: five panels sharing one pair of axes — the pooled base first, then one per segment — each captioned with its own correlation and customer count.

Input: the feature table and the Chapter 6 segment labels. Transformation: one pooled scatter and four conditional ones, on shared axes. Output: the always-plot discipline applied to the finance claim. Shared axes are the whole design: the panels are comparable only if the scales are identical, and a faceted exhibit with free axes is the most common way a mixed-population plot conceals the very structure it was drawn to reveal. Write one sentence per panel describing what the pooled plot concealed, and note in particular whether any segment's cloud occupies a region of discount_share that the other segments barely reach — because a coefficient estimated across the pooled base will be an average over regions where the segments do not overlap, which is Section 7.8's extrapolation warning arriving one section early and in visual form.

Code 7.7. Within-segment correlations against the pooled value

```
rows = []
for seg, g in cf_full.groupby("segment"):
    rows.append({
        "segment": seg,
        "customers": len(g),
        "r_discount_monetary": round(
            g["discount_share"].corr(g["monetary"]), 3),
        "r_app_monetary": round(g["app_user"].corr(g["monetary"]), 3),
        "mean_discount_share": round(g["discount_share"].mean(), 3),
        "mean_monetary": round(g["monetary"].mean(), 2),
    })
within = pd.DataFrame(rows).set_index("segment")

assert within["customers"].sum() == len(cf_full)

print("pooled discount-monetary r:",
      round(cf_full["discount_share"].corr(cf_full["monetary"]), 3))
print(within)
```

Expected output: the pooled correlation on one line, then one row per segment carrying its size, its two within-segment correlations, and the two segment means that explain the pooled figure.

Input: the feature table grouped by segment. Transformation: the same two correlations computed inside each group, beside the group means. Output: the confounding-by-composition exhibit and the finance claim's rebuttal. The two mean columns are why the cell prints them: if the segment with the highest mean discount_share also has the lowest mean monetary, the pooled negative correlation is substantially a statement about which segment a customer belongs to, and the within-segment correlations say what remains once that composition is set aside. Note the direction of the conclusion carefully. Weak or reversed within-segment correlations do not establish that discounts raise spending; they establish that the pooled correlation cannot be read as evidence that discounts lower it.

VERIFICATION CHECK

Before you run: predict the direction of the change. If segment composition is doing the work Section 7.5 claims, the pooled negative discount–spend correlation should weaken, vanish, or reverse within segments. Write the prediction down, and write the alternative outcome too: if the association survives at full strength inside every segment, the composition story explains less than claimed and that is equally a finding.

After you run: the pooled figure, the four within-segment figures, and the two mean columns are all in the notebook, and one insight statement per Section 5.9 records the comparison, its magnitude, its connection to the budget decision, and a verb that stops at the evidence. The whale comparison from Code 7.5 sits beside it, so that a correlation that both depends on the top decile and dissolves within segments is flagged twice rather than once.

Investigate if: a within-segment correlation is computed on a segment with very few customers, or on a segment whose discount_share barely varies. Both produce unstable correlations that look like findings, and the customers column and the faceted plot from Code 7.6 are where to check. A correlation inside a group whose predictor has almost no spread is arithmetic noise, not a weak relationship — and in the limiting case, where a segment's discount_share takes a single value, r is not small but undefined, since its denominator is zero. Software may report NaN there; read it as "the question cannot be asked of this group," not as "no relationship."

7.12.3 Lab 7.2, Part A: The Composition Features and the Anchor Model

Lab 7.2 builds the deliverable in four parts, moving from one predictor to the full specified model. The response variables are monetary (total spend) and aov, both defined per Table 6.5. Two additional features join from the certified transactions file, built with Chapter 4's groupby mechanics and reconciled before use: units_per_order (total units ÷ distinct orders) and occasionwear_share (revenue on occasionwear lines ÷ total revenue, joining products for the is_occasionwear flag). Declare both derivation rules in the notebook before computing them.

Code 7.8. Build and reconcile the two composition features

```
analysis_date = pd.Timestamp("2026-06-30")
window_start = analysis_date - pd.DateOffset(months=24)

# Certified figures for the DECLARED WINDOW, copied from the Chapter 4
# verification log before anything is computed. Fill these in from your
# own log: a build that reconciles only against itself cannot catch an
# error it inherited.
CERT_REVENUE    = ...   # total revenue in the window
CERT_PURCHASERS = ...   # distinct customers with an order in the window

window = transactions_clean[
    (transactions_clean["order_date"] > window_start)
    & (transactions_clean["order_date"] <= analysis_date)]

# The occasionwear flag lives on the product dimension, so the join is
# validated: many lines to one product, and nothing may go unmatched.
lines = window.merge(products[["product_id", "is_occasionwear"]],
                     on="product_id", how="left", validate="many_to_one")
assert len(lines) == len(window), "the join changed the row count"
assert lines["is_occasionwear"].notna().all(), "unmatched product_id"

comp = (lines
        .assign(occasion_revenue=lambda d: d["line_revenue"]
```

```
                .where(d["is_occasionwear"], 0.0))
        .groupby("customer_id")
        .agg(units=("quantity", "sum"),
             orders=("order_id", "nunique"),
             revenue=("line_revenue", "sum"),
             occasion_revenue=("occasion_revenue", "sum")))

comp["units_per_order"] = comp["units"] / comp["orders"]
comp["occasionwear_share"] = comp["occasion_revenue"] / comp["revenue"]

model_data = cf_full.join(
    comp[["units_per_order", "occasionwear_share"]], how="left")

# Reconciliation: a feature built from the same window on the same file
# must agree with the certified feature table, or one of them is wrong.
assert len(model_data) == len(cf_full)
assert (comp["orders"].reindex(cf_full.index) == cf_full["frequency"]).all()
assert np.allclose(comp["revenue"].reindex(cf_full.index),
                   cf_full["monetary"], atol=0.01)
assert len(cf_full) == CERT_PURCHASERS
assert np.isclose(comp["revenue"].sum(), CERT_REVENUE, atol=0.01)

tol = 1e-9
assert model_data[["units_per_order",
                   "occasionwear_share"]].notna().all().all()
assert (model_data["units_per_order"] >= 1 - tol).all()
assert model_data["occasionwear_share"].between(-tol, 1 + tol).all()

print(model_data[["units_per_order", "occasionwear_share"]]
      .describe().round(3))
```

Expected output: an eight-row summary for the two new features, with units_per_order at or above 1.0 throughout and occasionwear_share bounded by 0 and 1, and nine assertions passing.

Input: the certified line-grain file and the product dimension. Transformation: a validated join, a conditional revenue column, one grouped aggregation, and two ratios. Output: the two composition features Chapter 5's decomposition named, joined to the modeling table. The reconciliation assertions are the point of the cell, and they run at two levels. The first pair recomputes orders and revenue per customer from the same window that produced the certified feature table, so the two must agree exactly. The second pair checks both against the Chapter 4 verification log, because a build that reconciles only against the feature table will happily reproduce an error the feature table already contains — the internal check catches this chapter's mistakes, and the certified check catches the ones inherited from upstream. The share needs a tolerance rather than a bare comparison with 1, for the reason Chapter 6 met: a share summed from rounded line revenues can land a fraction above one in floating point.

Note the ordering discipline the join enforces. The composition features are built from the transaction file, not from the feature table, because units and occasionwear status live at line grain and cannot be recovered from customer totals — the same grain argument that sent Chapter 6 back to transactions for discount_share. Building them here rather than in Chapter 6 is a deliberate scope decision: they exist to serve this chapter's average-order-value model, and a feature table should not accumulate columns for analyses nobody has specified.

Code 7.9. The anchor model, with nonrobust and robust standard errors side by side

```
import statsmodels.formula.api as smf

m1_default = smf.ols("monetary ~ recency_days", data=model_data).fit()
m1 = smf.ols("monetary ~ recency_days",
             data=model_data).fit(cov_type="HC3")

comparison = pd.DataFrame({
    "coefficient": m1.params.round(4),
```

```
    "SE (nonrobust)": m1_default.bse.round(4),
    "SE (HC3)": m1.bse.round(4),
})

# HC3 changes the estimated standard errors, confidence intervals, and
# p-values. It does NOT change the fitted coefficients or the predictions.
assert np.allclose(m1.params, m1_default.params)
assert np.isclose(m1.rsquared, m1_default.rsquared)
assert np.allclose(m1.fittedvalues, m1_default.fittedvalues)

print(comparison)
print("R-squared (identical under both):", round(m1.rsquared, 3))
print("\n95% interval on recency_days, HC3:")
print(m1.conf_int().loc["recency_days"].round(3))
print("95% interval on recency_days, nonrobust:")
print(m1_default.conf_int().loc["recency_days"].round(3))
```

Expected output: a two-row table whose coefficient column is identical under both fits and whose two standard-error columns differ; one R-squared value; and two intervals on the recency slope that share a center and differ in width.

Input: the modeling table. Transformation: the same regression fitted twice, under two covariance estimators. Output: the difference robust inference makes, isolated. Read the slope aloud in units — dollars of 24-month spend per additional day since last purchase — and write the interval sentence per Section 7.10. Then read the three assertions as the chapter's claim in executable form: the estimate, the fit, and every prediction are untouched by the choice of estimator, and everything used to judge the estimate's precision is not. An analyst who reports the coefficient from one fit and the p-value from the other has mixed two answers to two different questions.

Code 7.10. Residual diagnostics: the funnel, measured rather than eyeballed

```
fig, axes = plt.subplots(1, 2, figsize=(11, 4))

axes[0].scatter(m1.fittedvalues, m1.resid, s=4, alpha=0.25)
axes[0].axhline(0, linewidth=0.8)
axes[0].set_xlabel("fitted spend ($)")
axes[0].set_ylabel("residual ($)")
axes[0].set_title("Residuals against fitted values")

for seg in segments:
    mask = (model_data["segment"] == seg).values
    axes[1].scatter(m1.fittedvalues[mask], m1.resid[mask],
                    s=4, alpha=0.4, label=seg)
axes[1].axhline(0, linewidth=0.8)
axes[1].set_xlabel("fitted spend ($)")
axes[1].set_title("The same residuals, colored by segment")
axes[1].legend(fontsize=7, markerscale=2)

plt.tight_layout()
plt.show()

# A funnel is a claim about spread, so measure the spread. Residual
# standard deviation by decile of fitted value turns the picture into a
# number the Verification Check can be written against.
spread = (pd.DataFrame({"fitted": m1.fittedvalues, "resid": m1.resid})
          .assign(decile=lambda d: pd.qcut(d["fitted"], 10, labels=False,
                                           duplicates="drop"))
          .groupby("decile")["resid"]
          .agg(residual_sd="std", n="size")
          .round(2))
print(spread)
```

Expected output: two residual plots sharing a scale, and a ten-row table in which the residual standard deviation rises materially from the lowest fitted decile to the highest.

Input: the fitted anchor model. Transformation: two renderings and one grouped spread calculation. Output: the diagnostic evidence that justifies both the robust estimator already used and the fuller model about to be fitted. The left panel is the classic exhibit; the right panel is the one that changes the specification, because visible clustering by segment is the data arguing that four groups do not share one line. The decile table converts an impression into a measurement, which matters for two reasons: an eyeballed funnel is exactly the kind of judgment that quietly disappears when the analysis is delegated, and a rising residual standard deviation is the concrete fact that makes the case for HC3 in a sentence a non-technical reader can follow.

7.12.4 Lab 7.2, Part B: The Specified Spend Model

Part B fits the specified spend model: monetary on recency_days, tenure_days, app_user, email_opt_in, and discount_share, with the four segments entering as dummies against the Urban Loyal Core reference per Table 7.3. Before running, write the predicted sign of every coefficient and one anchor magnitude — the raw app-user spend gap, which Code 7.11 prints beside the model so the shrinkage can be read directly.

Code 7.11. Segment dummies and the specified spend model

```
# Table 7.3 as code. The reference category gets no column: it is
# represented when all three indicators are zero.
SEG_REF = "Urban Loyal Core"
SEG_CODES = {
    "Suburban Occasion": "seg_suburban",
    "Discount-Dependent": "seg_discount",
    "New / At-Risk": "seg_new",
}
for name, col in SEG_CODES.items():
    model_data[col] = (model_data["segment"] == name).astype(int)

seg_cols = list(SEG_CODES.values())
# Exactly one indicator on for a non-reference customer, none for the
# reference: the encoding is exhaustive and mutually exclusive.
assert model_data[seg_cols].sum(axis=1).isin([0, 1]).all()
assert (model_data[seg_cols].sum(axis=1) == 0).sum() == (
    model_data["segment"] == SEG_REF).sum()

spend_formula = (
    "monetary ~ recency_days + tenure_days + app_user + email_opt_in"
    " + discount_share + seg_suburban + seg_discount + seg_new")

m2 = smf.ols(spend_formula, data=model_data).fit(cov_type="HC3")
print(m2.summary().tables[1])
print("R-squared:", round(m2.rsquared, 3))

# The anchor magnitude: the raw gap the platform deck quoted.
raw = model_data.groupby("app_user")["monetary"].mean()
print(f"\nraw app-user spend gap:   ${raw.loc[1] - raw.loc[0]:,.2f}")
print(f"raw app-user spend ratio: {raw.loc[1] / raw.loc[0]:.2f}x")
print(f"adjusted app coefficient: ${m2.params['app_user']:,.2f}")
```

Expected output: a coefficient table with nine rows including the three segment indicators; an R-squared value; and three lines contrasting the raw app-user gap and ratio with the adjusted coefficient, the adjusted figure being substantially the smaller.

Input: the modeling table. Transformation: three indicator columns and one fitted model. Output: the drivers table the meeting will quote, and the single comparison that disciplines it. The two assertions make Table 7.3's encoding executable rather than merely described: every non-reference customer carries exactly one indicator, and

the count of all-zero rows equals the reference segment's size. That second assertion is the one that catches a mis-specified reference — the most common silent error in dummy coding, and one that changes every printed number without changing the model's fit.

Read the app comparison in the order printed. The raw ratio is a difference between two groups of customers who differ in everything that travels with app ownership. The adjusted coefficient is a model-based comparison holding recency, tenure, email engagement, discount exposure, and segment membership constant — and the gap between the two numbers is the chapter's central exhibit, because it prices how much of a widely quoted business fact is profile rather than product.

Code 7.12. The coefficient table as a deliverable: estimates, intervals, and units

```
UNITS = {
    "Intercept": "dollars, at zero on every predictor",
    "recency_days": "dollars per additional day since last purchase",
    "tenure_days": "dollars per additional day of tenure",
    "app_user": "dollars, app users vs. non-users",
    "email_opt_in": "dollars, opted-in vs. not",
    "discount_share": "dollars per 1.00 increase in discount share",
    "seg_suburban": "dollars vs. the Urban Loyal Core",
    "seg_discount": "dollars vs. the Urban Loyal Core",
    "seg_new": "dollars vs. the Urban Loyal Core",
}

def coefficient_table(fit, units=None, digits=2):
    """A coefficient table a meeting can read: estimate, interval, units."""
    low, high = fit.conf_int()[0], fit.conf_int()[1]
    out = pd.DataFrame({
        "estimate": fit.params.round(digits),
        "ci_low": low.round(digits),
        "ci_high": high.round(digits),
        "interval_width": (high - low).round(digits),
        "p_value": fit.pvalues.round(4),
    })
    if units:
        out["reads as"] = [units.get(i, "") for i in out.index]
    return out

table = coefficient_table(m2, UNITS)

# An interval that does not contain its own estimate is a construction bug.
assert (table["ci_low"] <= table["estimate"]).all()
assert (table["estimate"] <= table["ci_high"]).all()

print(table)
```

Expected output: a nine-row table carrying, for every coefficient, its estimate, its HC3 interval, the interval's width, its p-value, and a plain-language units string.

Input: the fitted model. Transformation: a reshaping of the results object. Output: the exhibit that goes in the deliverable. The units column is the cell's argument. Section 7.6 required that a coefficient not be read until its units are said aloud, and a units dictionary written by hand before the table is printed forces the analyst to say them once for every predictor — including the two that trip people most reliably. A dummy's units are dollars relative to the reference, not dollars absolutely. And discount_share is a proportion, so its coefficient describes a difference of one full unit of share, which is the entire range from no discounted revenue to all of it; quoting it as though it were a percentage point is a hundred-fold error that no software will catch.

The interval width column earns its place for the reason Section 7.10 gave: at this sample size nearly everything clears conventional significance, so the p-value column separates almost nothing and the width column separates a

great deal. Sort the table by width and the coefficients the analysis actually pins down separate themselves from the ones it merely signs.

Code 7.13. Variance inflation factors

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from patsy import dmatrix

# VIF is computed on the model's design matrix, including the intercept
# column, and then reported for the predictors only.
X = dmatrix(spend_formula.split("~", 1)[1], data=model_data,
            return_type="dataframe")

vif = pd.Series(
    [variance_inflation_factor(X.values, i) for i in range(X.shape[1])],
    index=X.columns, name="VIF").drop("Intercept")

assert (vif >= 1 - 1e-6).all(), (
    "a VIF below 1 means the design matrix was built wrong")

print(vif.sort_values(ascending=False).round(2))
print("\npredictor correlations, for comparison:")
print(model_data[["recency_days", "tenure_days", "app_user",
                  "email_opt_in", "discount_share"]].corr().round(2))
```

Expected output: eight variance inflation factors sorted from largest to smallest, each at least 1, followed by the five-by-five correlation matrix of the numeric and indicator predictors.

Input: the specified model's design matrix. Transformation: one variance inflation factor per predictor. Output: the multicollinearity diagnostic that pairwise correlations cannot supply. Print the two together, as the cell does, because the comparison is the lesson: a predictor can show no alarming pairwise correlation and still carry a raised variance inflation factor, because the factor measures its relationship with all the other predictors at once. The three segment indicators are the natural place to see it, since they are mutually exclusive by construction and therefore related to one another whatever their pairwise correlations look like.

Resist the cutoff. This guide publishes none, and the reason is practical rather than pedantic: the number that matters is whether a particular coefficient is estimated precisely enough for the decision in front of you, which is a question the interval width in Code 7.12 answers directly and a threshold answers only by proxy. Read a large factor as an instruction — check this coefficient's stability across specifications, and consider reporting it jointly with its entangled partners rather than separately — which is exactly what Code 7.17 does.

> **VERIFICATION CHECK**
>
> Before you run: this is the four-point audit of Table 7.5, run on your own model before it is ever run on an assistant's. Write the predicted sign of all eight coefficients. Compute the raw app-user gap by hand from a two-line groupby and predict whether the adjusted coefficient will land above, near, or far below it. Predict the direction of the three segment coefficients against Chapter 5's certified descriptive gaps, and predict which two predictors will carry the largest variance inflation factors.

After you run: signs first — every coefficient against your written predictions, with any surprise earning a confounding story or a flag rather than a shrug. Magnitudes next: compare the adjusted segment coefficients with the certified descriptive gaps for sign, approximate scale, and substantive plausibility. An adjusted difference may come out smaller or larger than the raw difference; a larger one is not evidence of an error, because opposing covariate differences can suppress part of an association until the model removes them. Either direction requires an explanation grounded in the variables that entered the model. Then units: restate three coefficients aloud, including one dummy and discount_share. Then diagnostics: the residual funnel should be reduced relative to Code 7.10 but not gone, the variance inflation factors should be read, and the interval widths should be the basis of any claim about precision.

Investigate if: a segment coefficient exceeds its descriptive gap by a wide margin and no covariate story explains it, or the app coefficient fails to shrink at all against the raw gap. The first is worth tracing through Code 7.17's specification sensitivity; the second usually means a control that was supposed to carry the profile — segment membership most often — is absent, misspelled in the formula, or constant within the sample.

7.12.5 Lab 7.2, Part C: Average Order Value and Chapter 5's Decomposition

Part C turns to average order value, and to the chapter's designed payoff: a certified descriptive finding, remeasured by a new instrument. Chapter 5 decomposed the suburban average-order-value premium into basket size and occasionwear mix. Two models reproduce that finding in regression form — one with the segments alone, one with the two composition features added — and the comparison between them is the deliverable.

Code 7.14. The segment-only and composition-controlled AOV models

```
# The certified descriptive gap from the Chapter 5 verification log --
# suburban mean AOV minus Urban Loyal Core mean AOV, in dollars.
CERT_AOV_GAP_SUBURBAN = ...

aov_simple = smf.ols(
    "aov ~ seg_suburban + seg_discount + seg_new",
    data=model_data).fit(cov_type="HC3")

aov_controlled = smf.ols(
    "aov ~ seg_suburban + seg_discount + seg_new"
    " + units_per_order + occasionwear_share",
    data=model_data).fit(cov_type="HC3")

# A segment-only model IS the difference in group means. Checking that
# identity is how you know the encoding and the reference are right.
means = model_data.groupby("segment")["aov"].mean()
descriptive_gap = means["Suburban Occasion"] - means[SEG_REF]
assert np.isclose(aov_simple.params["seg_suburban"],
                  descriptive_gap, atol=0.01)
assert np.isclose(aov_simple.params["Intercept"], means[SEG_REF], atol=0.01)
# ...and the recomputed gap must match the certified one.
assert np.isclose(descriptive_gap, CERT_AOV_GAP_SUBURBAN, atol=0.01)

s_lo, s_hi = aov_simple.conf_int().loc["seg_suburban"]
c_lo, c_hi = aov_controlled.conf_int().loc["seg_suburban"]

print(f"descriptive AOV gap (Chapter 5): ${descriptive_gap:,.2f}")
print(f"segment-only coefficient:
${aov_simple.params['seg_suburban']:,.2f} "
```

```
      f"[{s_lo:,.2f}, {s_hi:,.2f}]")
print(f"with composition controls:
${aov_controlled.params['seg_suburban']:,.2f} "
      f"[{c_lo:,.2f}, {c_hi:,.2f}]")
# NOT a share of the gap: this is the proportional change in ONE
# coefficient between two specifications. It can exceed 100% or go
# negative if the coefficient crosses zero.
proportional_reduction = (
    1 - aov_controlled.params["seg_suburban"]
      / aov_simple.params["seg_suburban"]
)
print(f"\nproportional reduction in the suburban coefficient "
      f"after adding the composition variables: "
      f"{proportional_reduction:.1%}")
print(f"R-squared, segment only: {aov_simple.rsquared:.3f}")
print(f"R-squared, controlled:   {aov_controlled.rsquared:.3f}")
```

Expected output: the descriptive gap and the segment-only coefficient agreeing to the cent, a controlled coefficient much closer to zero with an interval that may include it, the proportional reduction in the suburban coefficient, two R-squared values, and three assertions passing.

Input: the modeling table with its composition features. Transformation: two fitted models sharing a reference category. Output: Chapter 5's decomposition in coefficient form. The three assertions are the cell's verification spine, and they are worth pausing on. The third is the one that reaches outside this chapter: the gap the model reproduces must equal the gap Chapter 5 certified, so a segment label that drifted or a population that changed is caught here rather than in a meeting. The first two are worth pausing on for a different reason. A regression of a response on a complete set of category indicators with an intercept reproduces the group means exactly: the intercept is the reference group's mean, and each indicator's coefficient is that group's mean minus the reference's. In a model containing nothing else, therefore, the intercept and the reference mean coincide — the special case Section 7.9's definition named. Add the two composition variables and the intercept stops being anybody's mean, which is the general case the definition insisted on.

The wording of the finding is where this part earns its place in the chapter, and the temptation is specific. It is tempting to write that the controlled model shows the suburban premium runs through composition. That sentence describes a causal pathway — a mediation claim — and the two fitted models do not establish one. What the evidence supports is a statement about adjustment: after adjusting for units per order and occasionwear share, the estimated suburban difference becomes much smaller, so these variables statistically account for most of the observed average-order-value gap in this dataset. That formulation agrees with Chapter 5 exactly, survives a CFO's follow-up question, and does not claim a mechanism the analysis has not tested.

7.12.6 Lab 7.2, Part D: Interactions, Consistency, Sensitivity, and Leakage

Part D estimates the dataset's designed interaction, tests a claim the managerial section would otherwise have made on faith, and closes with the two audits that keep a coefficient table honest: a specification sweep and a deliberate leakage demonstration.

Code 7.15. The loyalty-by-app interaction and its two conditional associations

```
# .ne("None") treats a missing tier as NOT equal to "None", which
# would silently file unknown customers as enrolled. Resolve first.
assert model_data["loyalty_tier"].notna().all(), (
    "resolve missing loyalty_tier values before defining enrollment"
)
model_data["loyalty_enrolled"] = (
    model_data["loyalty_tier"].ne("None").astype(int))

# An interaction estimates a separate association per group, so check the
# group sizes BEFORE fitting. Empty or thin cells produce a coefficient
# with an interval wide enough to contain any story anyone wants.
```

```
cells = pd.crosstab(model_data["loyalty_enrolled"], model_data["app_user"])
print(cells)
MIN_CELL = 30  # course screening threshold, not a universal guarantee
assert (cells.values >= MIN_CELL).all(), (
    "one interaction cell falls below the course screening threshold"
)

interaction = smf.ols(
    "monetary ~ app_user * loyalty_enrolled + recency_days + tenure_days"
    " + discount_share + seg_suburban + seg_discount + seg_new",
    data=model_data).fit(cov_type="HC3")
print(interaction.summary().tables[1])

def linear_combination(fit, terms, label):
    """Estimate and interval for a sum of coefficients (Section 7.9)."""
    contrast = np.zeros(len(fit.params))
    for t in terms:
        contrast[list(fit.params.index).index(t)] = 1.0
    test = fit.t_test(contrast)
    est = float(np.ravel(test.effect)[0])
    lo, hi = np.ravel(test.conf_int())[:2]
    print(f"{label}: ${est:,.2f}  [{lo:,.2f}, {hi:,.2f}]")
    return est

not_enrolled = linear_combination(
    interaction, ["app_user"],
    "app association, NOT loyalty-enrolled")
enrolled = linear_combination(
    interaction, ["app_user", "app_user:loyalty_enrolled"],
    "app association, loyalty-enrolled    ")

# The conditional reading of Section 7.9, made executable.
assert np.isclose(enrolled, interaction.params["app_user"]
                  + interaction.params["app_user:loyalty_enrolled"])
```

Expected output: a two-by-two table of cell counts, a coefficient table containing app_user, loyalty_enrolled, and app_user:loyalty_enrolled, then two conditional app associations in dollars with their HC3 intervals, and three assertions passing.

Input: the modeling table. Transformation: one derived indicator, one cell-count check, one fitted model with a product term, and two linear combinations. Output: the two numbers the meeting can act on. Two guards run before the model. The first is a missing-value check on loyalty_tier, and it exists because the derivation immediately below it is booby-trapped: .ne("None") evaluates a missing tier as not equal to "None", so any customer whose tier never arrived would be filed as enrolled without a word of complaint — Chapter 4's two-kinds-of-null problem reappearing as a silent recoding. The second is the cell-count check, and it is not boilerplate either: an interaction asks the data for a separate association in each combination of two variables, and a combination holding a handful of customers will supply one, complete with a confident-looking point estimate and an interval nobody reads. Read the threshold for what it is. Thirty is this course's screening floor, chosen so a lab cell has enough customers to estimate a slope at all; it is not a statistical sufficiency boundary, and clearing it guarantees nothing about precision. The confidence intervals printed below still decide what the sample actually supports, and a cell that passes the screen can easily produce an interval too wide to act on.

The two printed associations are the deliverable, and the interaction coefficient itself is not. A meeting sentence built on "the interaction was significant" tells a VP nothing actionable; two dollar figures with intervals — the app association among non-enrolled customers, and the app association among enrolled customers — tell the four journeys something they can be designed around. Note also what the conditional reading does to the main-effect row in the printed table: app_user is now the association among the non-enrolled only, and quoting it as the overall app association would be exactly the misreading Table 7.5's third point exists to catch.

Code 7.16. Is the email association the same in every segment?

```
email_formula = (
    "monetary ~ email_opt_in + seg_suburban + seg_discount + seg_new"
    " + email_opt_in:seg_suburban + email_opt_in:seg_discount"
    " + email_opt_in:seg_new"
    " + recency_days + tenure_days + app_user + discount_share")

email_model = smf.ols(email_formula, data=model_data).fit(cov_type="HC3")

# One joint test of the three differences, then the four conditional
# associations. The joint test asks whether the associations differ at
# all; the four numbers say what they are.
diff_terms = [p for p in email_model.params.index
              if p.startswith("email_opt_in:")]
R = np.zeros((len(diff_terms), len(email_model.params)))
for row, term in enumerate(diff_terms):
    R[row, list(email_model.params.index).index(term)] = 1.0
joint = email_model.f_test(R)
print("joint test that all three email-by-segment differences are zero:")
print(f"  F = {float(np.ravel(joint.fvalue)[0]):.2f}, "
      f"p = {float(np.ravel(joint.pvalue)[0]):.4f}")
print()

for seg, col in [(SEG_REF, None)] + list(SEG_CODES.items()):
    terms = ["email_opt_in"]
    if col:
        terms.append(f"email_opt_in:{col}")
    linear_combination(email_model, terms, f"email association, {seg:<20}")
```

Expected output: one joint F test with its p-value, then four conditional email associations in dollars with HC3 intervals — one per segment, the first being the Urban Loyal Core reference.

Input: the modeling table. Transformation: a model carrying one interaction per non-reference segment, one joint test, and four linear combinations. Output: the evidence behind a managerial sentence that would otherwise be an assumption. This cell exists because of a specific discipline failure worth naming. A draft of this chapter's managerial section reported that the email association is consistent across segments — a claim about four quantities that no model in the chapter had estimated. Either the analysis is run or the claim is withdrawn, and running it is cheap.

Read the two outputs in order and let them do different jobs. The joint test asks a single question: are the three segment differences all zero? A large p-value there does not establish that the associations are identical — failing to detect a difference is not evidence of sameness, particularly where the segments differ in size and the smallest ones carry the widest intervals. The four conditional estimates are what the deliverable reports, and the honest summary names both the common direction and the spread of the intervals around it. Section 7.13 writes that sentence.

Code 7.17. Specification sensitivity: how much does the app coefficient depend on the model?

```
SPECS = {
    "A. app only": "monetary ~ app_user",
    "B. + behavior, no segments":
        "monetary ~ app_user + recency_days + tenure_days"
        " + email_opt_in + discount_share",
    "C. the specified model": spend_formula,
    "D. + purchase intensity":
        spend_formula + " + orders_per_month",
}

rows = []
for name, formula in SPECS.items():
```

```
    fit = smf.ols(formula, data=model_data).fit(cov_type="HC3")
    lo, hi = fit.conf_int().loc["app_user"]
    rows.append({
        "specification": name,
        "app_coefficient": round(fit.params["app_user"], 2),
        "ci_low": round(lo, 2),
        "ci_high": round(hi, 2),
        "r_squared": round(fit.rsquared, 3),
        "predictors": int(fit.df_model),
    })

sensitivity = pd.DataFrame(rows).set_index("specification")

# R-squared cannot fall as predictors are added to a NESTED model on the
# same rows (Section 7.7). A and B and C are nested; C and D are nested.
assert sensitivity.loc["C. the specified model", "r_squared"] >= \
       sensitivity.loc["B. + behavior, no segments", "r_squared"]
assert sensitivity.loc["D. + purchase intensity", "r_squared"] >= \
       sensitivity.loc["C. the specified model", "r_squared"]

print(sensitivity)
```

Expected output: four rows showing the app coefficient shrinking substantially from specification A to specification C, a further movement at D, non-decreasing R-squared down the nested sequence, and two assertions passing.

Input: the modeling table. Transformation: four nested fits, each reporting the same coefficient. Output: the instability report that turns a single number into a range of defensible numbers. This is the specification-sensitivity check Section 7.10 named as multicollinearity's paired instrument, and it is also the honest answer to the question every drivers deliverable invites: how much does this depend on your choices? Specification D is the pointed one, because orders_per_month is close kin to the response and to several predictors at once; watch what it does to the app coefficient's interval as well as to its point estimate, and note that a rising R-squared accompanies it either way.

The two assertions encode Section 7.7's conditions rather than a general law. R-squared cannot fall when predictors are added to a nested model fitted to the same observations — which is why the comparison being asserted is B against C and C against D, and why specification A, which is nested inside all of them, is left to the eye. If either assertion ever fails, the models are not fitted on the same rows, and a silently dropped observation is the first place to look.

Code 7.18. The leakage audit: a model handed its own answer

```
# Chapter 6 defined aov = monetary / frequency (Table 6.5). Their product
# is therefore monetary itself, exactly. Predict both R-squared values
# before running this cell.
additive = smf.ols("monetary ~ aov + frequency", data=model_data).fit()
product = smf.ols("monetary ~ I(aov * frequency)",
                  data=model_data).fit()

print("additive model, monetary ~ aov + frequency")
print("  R-squared:", round(additive.rsquared, 4))
print("product model, monetary ~ I(aov * frequency)")
print("  R-squared:", round(product.rsquared, 8))

identity_error = (model_data["aov"] * model_data["frequency"]
                  - model_data["monetary"]).abs().max()
print("\nlargest |aov x frequency - monetary|:", f"{identity_error:.2e}")

# The additive model fits strongly and NOT perfectly: a sum is not a
# product. The product model reconstructs the response by construction.
assert additive.rsquared < 0.999
```

```
assert product.rsquared > 0.9999
assert identity_error < 1e-6
```

Expected output: a strong but imperfect R-squared for the additive model, an R-squared of essentially 1 for the product model, an identity error at floating-point scale, and three assertions passing.

Input: the modeling table. Transformation: two regressions of the response on its own components. Output: the sensation of a model that is too good, produced on purpose. The two specifications are here together because the difference between them is the lesson. Adding aov and frequency as separate additive predictors fits strongly — they are the response's own factors — but it does not reconstruct the response, because a linear combination of two variables is not their product. Multiply them and the model is handed monetary under another name, and R-squared goes to one up to floating-point error.

Name what happened in the notebook, in the words the deliverable would use: the too-good model did not discover customer behavior; it was handed the outcome under another name. Then generalize the detection rule, because the next instance will not be this obvious. Trace every predictor's derivation before it enters a model, exactly as Table 7.2's fourth row instructed for correlations; treat any R-squared above roughly 0.9 on individual customer behavior as a defect report rather than a result; and remember that the identity holds here to floating-point precision only because Chapter 6's build does not round aov — a version that rounded to cents would produce an R-squared microscopically below one, which is a fine illustration of how nearly invisible a leaking variable can be. Chapter 8 will name the general disease and equip you to detect it where no arithmetic identity announces it.

VERIFICATION CHECK

Before you run: predict both R-squared values in Code 7.18, in writing, before executing it — the additive model and the product model. Predict the two conditional app associations of Code 7.15 in dollars: what should the app–spend association be among customers who are not loyalty-enrolled, and how should it differ among the enrolled if loyalty and app engagement reinforce each other as the dataset's specification designed? For Code 7.16, predict whether the joint test will detect differences across segments, and say what you would conclude from either answer before you know which one you get.

After you run: the two conditional associations are written in dollars with their intervals, one sentence each, and the interaction coefficient is not quoted on its own anywhere in the deliverable. The email section reports four conditional estimates and the joint test, with a sentence that does not convert a large p-value into a claim of sameness. The specification table shows the app coefficient's range across four defensible models, and the deliverable quotes that range rather than a single number wherever the choice of specification moves it materially.

Investigate if: the product model's R-squared is not essentially 1. Either aov was rounded somewhere upstream, in which case the identity error printed by the cell will be at cents scale rather than floating-point scale, or the two columns are not the ones Table 6.5 defines. Both are derivation questions, and both are answered in the feature table rather than in the model.

Close Lab 7.2 with the AI round trip. Give an assistant the raw feature table and the deliberately loose prompt "find what drives spend and tell me what to do," then run Table 7.5's audit on the reply, grade its verbs against Section 7.2's registry, and file the exchange per Appendix D. The comparison between the assistant's narration and your Part B through Part D findings — same data, same arithmetic, different discipline — is the lab's last exhibit and the chapter's argument in miniature. Deliverables for both labs: the notebook with all eighteen cells executed, the written predictions made before each model was fitted, the reconciliation output from Code 7.8, the coefficient table with intervals and units, the specification-sensitivity table, the two conditional interaction associations in dollars, and the AI-Use Appendix documenting every assistant exchange, including at least one delegated step you corrected and why.

7.13 Marketing Interpretation and Managerial Insight

The lab's outputs are coefficient tables; the budget lock runs on sentences. This section translates — and, per this guide's standing practice, it does so partly by exhibiting a wrong managerial reading and correcting it, because this chapter's characteristic misreading is the most expensive one in the guide so far: a misread summary misinforms a meeting, a misread segmentation runs for a season, but a misread driver reallocates a budget and then defends the reallocation with the very analysis that caused it.

The wrong reading will be offered in the allocation meeting by someone reasonable, probably while pointing at your own Code 7.12 table: "The model says app users spend $140 more than comparable non-users. We have several thousand customers without the app. Fund the install campaign — at $140 a head, it pays for itself many times over." Every number in the sentence can be exactly right, and the conclusion does not follow. The coefficient is a model-adjusted comparison between existing app users and non-users; it is not a forecast of what happens when a non-user is persuaded to install. The customers who chose the app differ from those who did not in everything the model cannot see — enthusiasm for the brand, planned future purchases, the underlying engagement that Section 7.8's omitted-variable logic says is inflating the coefficient right now. Installing the app does not install the enthusiasm.

The corrected reading, which the deliverable should print under the table: "Customers with the app spend $140 more than otherwise similar customers without it, adjusting for recency, tenure, email engagement, discount exposure, and segment. Some of that difference may reflect what the app enables; some may reflect selection into app use, and this analysis cannot split the two. The raw gap the platform deck quoted was several times larger, and the model attributes most of it to profile rather than to the product. Across four defensible specifications the estimate ranges from $X to $Y, which is the range the decision should carry. If the install campaign is funded, it should be funded as a test — Chapter 11's instrument, not this chapter's — with a holdout that would tell us, one quarter from now, what a caused install is actually worth." That paragraph — association stated, rivals named, sensitivity disclosed, decisive evidence specified — is what an honest drivers deliverable sounds like, and it is a stronger meeting position than the overclaim, because it cannot be dismantled by the CFO's first question.

The same discipline settles the other two claims, and the deliverable should give each its verdict paragraph. The discount claim dissolves under the lab's within-segment exhibit from Code 7.7 and the faceted plot from Code 7.6: the pooled negative association reflects which customers use discounts more than what discounts do, and discount exposure is more strongly associated with the Discount-Dependent segment's purchasing pattern than with any change in it. Cutting the ladder on this evidence would withdraw the lever most visibly associated with a quarter of the treatment map's purchasing, on a correlation that composition substantially accounts for, and without any test of how that quarter responds. The recommendation that survives: hold the ladder, and test depth variations within the Discount-Dependent journey, where the lever's marginal effect is the live question.

The email claim earns the mildest verdict, and it is now a verdict the chapter has actually earned rather than assumed. Code 7.16 estimated the email association separately in each segment. Report what it found in the deliverable's own words: email engagement is positively associated with spend in every segment, in the specified model, with the four conditional estimates falling in a band the deliverable should print; the joint test of whether those associations differ is reported alongside them; and a large p-value on that test does not establish that the association is identical across segments, because failing to detect a difference is not evidence of sameness, especially where the smaller segments carry the widest intervals. What the chapter cannot supply from any of this is a price for additional sends — the day-grain version of the claim was disqualified by the calendar in Section 7.5, and no model here varies send frequency at all. Doubling frequency on this evidence would be paying twice for the same association; the honest sentence funds the content of the engaged journeys, not the volume, pending a frequency test.

The average-order-value finding gets a fourth paragraph and its own careful verb. After adjusting for units per order and occasionwear share, the estimated suburban difference becomes much smaller — these two variables statistically account for most of the observed premium in this dataset. Say it that way. The tempting version, that the premium runs through composition, asserts a pathway, and a pathway is a causal structure that two fitted models cannot establish. The adjustment claim is exactly as useful to the meeting, agrees with Chapter 5's certified decomposition, and does not require a retraction when someone asks how it was tested.

Two further translation disciplines round out the deliverable. First, magnitudes travel in decision units and with their intervals: "$0.85 less per day of recency, interval $0.70 to $1.00" prices a win-back trigger's timing; "recency was highly significant" prices nothing. Every quoted coefficient carries units, direction, the held-constant list in one clause, and its interval — the four-element metric discipline of Section 3.5.1, reborn for model outputs — and where the interval was computed with robust standard errors, which in this chapter is everywhere, the deliverable says so once in a methods line rather than in every sentence.

Second, the model's silences are reported alongside its statements: the drivers model says nothing about levers not in the data (creative quality, price perception, service), nothing about individuals (these are adjusted differences averaged over customers), nothing about combinations of predictor values the base barely contains, and nothing about the future. The one-page deliverable for the VP therefore has a fixed anatomy: the four verdict paragraphs; the coefficient table with intervals, in units; the composition exhibits that discipline the discount claim; the two conditional app associations from the interaction, in dollars, because that is the finding the four journeys can act on this fall; the specification-sensitivity range for any coefficient the recommendation leans on; and a closing line that names the next instrument — the questions this analysis cannot answer are precisely the ones an experiment can, and the cheapest of those experiments should be in the field before the next budget cycle. Drivers analysis, honestly delivered, does not end arguments; it upgrades them.

7.14 Business Analytics in Practice

This section turns from the fictional case to three ways regression is used — and misread — in professional marketing analytics, where it is at once the most institutionalized method in the field and the one whose outputs are most routinely pushed past what they can bear. The second and third vignettes are composites of recurring professional patterns rather than reports about a single named organization.

7.14.1 Marketing Mix Modeling

Marketing mix modeling is regression's flagship deployment and one of the oldest continuous analytics practices in the industry. It descends directly from the marketing-mix concept (Borden, 1964): regress a brand's sales, usually weekly and by market, on its marketing inputs — media spend by channel, price, promotion, distribution — plus the confounders this chapter taught you to name first: seasonality, holidays, competitor activity, macro conditions. The fitted coefficients become the currency of budget season, translated into contribution charts and response curves that answer, channel by channel, what a marginal dollar returned. For decades the method lived mostly at large consumer-goods firms with the data and patience for it; the privacy era revived it broadly, because as cookie deprecation, platform walled gardens, and tracking limits degraded user-level attribution, an approach that needs no individual tracking — aggregate spend in, aggregate sales out — became newly attractive.

Practitioners at the center of that revival have been candid about the method's fragilities, and the list reads like this chapter's table of contents at industrial scale: limited variation in the spend data, strong correlation among channels that are planned together, selection effects when spend chases demand, and results acutely sensitive to specification (Chan & Perry, 2017). Three of those deserve unpacking, because they are the difference between a model that is fitted and a model that is governed.

The first is that the specification is governed rather than merely chosen. In a mature practice the variable list, the functional forms for diminishing returns and carryover, the treatment of price and distribution, and the set of control variables are agreed in advance, documented, and changed through a review process — because a coefficient that moves when a control is added is a coefficient whose value is partly an artifact of a modeling choice, and an ungoverned specification lets that choice be made, unrecorded, by whoever ran the model last. The second is the specific fragility of the time controls. Seasonality, holiday timing, and the promotion calendar are the confounders that matter most, and they are also the ones most entangled with the media plan, since campaigns are scheduled into exactly the weeks when demand is highest; a model with too coarse a time control credits the season to the media, and a model with too fine a time control absorbs the media effect into the calendar. Neither error announces itself in the fit statistics. The third is collinearity among channels: media budgets move together, up in the fourth quarter and

down in the first, so the model is asked to separate contributions that the data present as a bundle — Section 7.10's problem, with a budget attached to its answer.

One distinction governs how the output may be used, and it is this chapter's distinction wearing an industrial suit. A marketing mix model used for historical attribution answers a descriptive question: given what we spent and what happened, how does the model apportion the sales we observed? A claim that moving next year's budget from one channel to another will cause a particular return is a different claim, resting on the assumption that the fitted relationship would survive an intervention that changes the very spend pattern the model was estimated on. Careful teams state which of the two they are doing, and increasingly they pay for the upgrade the same way this chapter recommends: deliberate spend experiments — geographic holdouts, staged rollouts — whose results calibrate and correct the regression. The practice lesson: the world's most consequential marketing regressions survive not because the fitting is clever but because the checking is institutional.

7.14.2 Confounding in the Wild

The second vignette is a composite of a pattern retail analytics teams meet regularly. A national retailer's team estimated discount elasticity from two years of transaction data: how much does volume respond when price is cut? The first model said, absurdly, that discounts barely mattered — and a specification review found the reason. Promotions clustered in December, when volume would have surged anyway; and the deepest discounts were routinely applied to the slowest merchandise, so discount depth was correlated with weak demand by managerial construction. The calendar pushed the estimate up; the clearance policy pushed it down; the two biases nearly canceled into a number that looked precise and meant nothing.

The repaired analysis compared like with like — within-season, within-category, promotion versus non-promotion weeks for the same items — and the elasticity that emerged was several times larger and finally decision-grade. The lesson is Section 7.5 operating at line-item scale: before trusting any observational coefficient, ask who chose the variable's values and why — because in business data the treatment is almost never assigned at random; it is assigned by someone's policy, and the policy is the confounder. It is also a lesson about the danger of two biases in opposite directions, since a badly confounded estimate does not always look wrong. Sometimes it looks reasonable, which is worse.

7.14.3 Key Driver Analysis

The third vignette is key driver analysis, the regression genre most students will meet first: customer-satisfaction and brand trackers, where survey attributes — ease of use, value for money, feels like a brand for me — are regressed on overall satisfaction or intent, and the coefficients are relabeled impact scores and sorted into a bar chart. The genre is useful, because it prioritizes attention across dozens of attributes that no one can weigh by intuition. It is also where every pathology this chapter names tends to appear at once, which makes a driver deck a good exam for a new analyst.

The attributes are strongly intercorrelated, because a satisfied customer rates everything higher, so the individual impact scores are multicollinearity's unstable division of shared credit, per Section 7.10 — and the ranking that the bar chart implies is precisely the quantity that instability makes unreliable. The causal direction is asserted rather than established: satisfied customers may rate value higher because they are satisfied, which is the reverse-causation rival from Section 7.2 sitting in the survey instrument itself. And the standard chart's language — improving this attribute by one point will lift satisfaction by X — is Section 7.8's coefficient-as-lever fallacy printed as a recommendation. Careful practitioners keep the genre honest by reporting attribute groups rather than pretending to separate entangled twins, by triangulating stated importance against modeled association, and by writing the deck's verbs in association language. Which mostly means that the analyst in the room did to the vendor's regression exactly what Table 7.5 does to an assistant's.

7.14.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into one expectation: you will spend more meeting-room minutes defending regressions from misreading than building them. The mix-model budget chart, the elasticity estimate, the driver bar chart — each will arrive in your inbox with causal language already attached, and the marginal value you

add is rarely a better fit; it is the audit of signs, magnitudes, units, and diagnostics, the confounder named before the decision, the specification recorded so that next quarter's model is comparable to this one, and the verb corrected before the slide ships. The statistics of this chapter are a solved problem in every software stack on earth. The judgment — what the coefficient can and cannot carry — is the job.

7.15 Ethics, Causal Overclaiming in Stakeholder Language

The Business Analytics in Practice section described analysts defending regressions from misreading. This section examines the defense as an ethical obligation, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), and differential treatment (Section 6.16) — to the act this chapter adds: stating what moves what. The chapter's ethics is not a caution about a side effect. Overclaiming is the technique's characteristic temptation, because association analysis produces artifacts that look exactly like causal answers, and the analyst is the only person in the chain positioned to know the difference.

Begin with how overclaiming actually happens, because the mechanism is rarely a lie. It is drift. The analyst writes "app ownership is associated with $140 higher spend, holding recency, tenure, engagement, and segment constant." The deck version becomes "app users spend $140 more." The executive summary becomes "the app drives $140 per customer." The board slide becomes "app adoption is a $1M growth lever." Each rewrite is shorter, each was made in good faith by someone optimizing for clarity, and the final sentence claims something the analysis never measured. No one decided to deceive; the qualifications were simply the least memorable part of every handoff, and compression removed them in order. The drift is worst under exactly two conditions this chapter creates: when the finding is quantified, because numbers survive compression and caveats do not, and when the finding flatters an existing plan, because the app team was always going to propose the install campaign. AI assistance intensifies both — Section 7.11 showed that assistants narrate causally by default, which means the first draft of the deck sentence now often arrives pre-drifted, and the analyst's role shifts from author to editor of claims.

Why does it matter, beyond precision for its own sake? Because overclaimed drivers move real money onto weak evidence, and the costs land asymmetrically — the asymmetric-cost lens of Section 2.7, applied to language. Fund the install campaign on the drifted claim, and the cost of being wrong is a wasted budget and a metric that quietly fails to move, discovered quarters later and attributed to execution. Cut the discount ladder on the drifted claim, and the cost lands on the Discount-Dependent segment — a quarter of the treatment map loses the lever most associated with its purchasing, on evidence that never tested how it responds — and on the P&L in the holiday quarter. And there is a second-order cost, paid by the discipline itself: every quantified overclaim that later fails burns credibility that the next honest analysis must spend to be believed. The analyst's overclaim is borrowed against colleagues' future evidence.

The discipline this guide installs has three parts, all rehearsed in this chapter and none of them new machinery. First, the verb registry of Section 7.2, applied at the point of writing: "is associated with," "predicts," and "differs by" for this chapter's findings; "drives," "causes," and "will lift" reserved for evidence that earned them, with the earning stated. The registry is a writing rule, which means it is auditable — a reviewer can grade a deliverable's verbs against its methods in five minutes, and Exercise 7.7 does. Second, the drift defense: the analyst controls the source document, so the source document carries the qualification inside the sentence, not in a footnote — "app users spend $140 more than otherwise similar non-users, a difference this analysis cannot attribute to the app itself" survives one more round of compression than the same content footnoted. Anticipating compression is part of writing the sentence. Third, the escalation duty: when a drifted claim surfaces in a meeting — someone else's slide, quoting your number with an upgraded verb — the analyst of record corrects it, with the same professional obligation as correcting a wrong number. The correction is rarely comfortable and is the whole point of having an analyst of record: Chapter 1 defined the role as accountability for what the analysis actually establishes, and the verb is part of what it establishes.

One boundary keeps the discipline honest in the other direction, because refusing all causal language is its own failure — an analyst who answers every question with "correlation is not causation" has abandoned the field to whoever will overclaim. The evidence for a relationship comes in grades: raw association; association with

confounders named; association robust to controls and to specification, which is what Code 7.17 measures; association consistent with a designed mechanism; and, above them all, experimental evidence. Honest stakeholder language locates the finding on that ladder and states what would move it up a rung: "this is a controlled association that holds across four specifications; the December test would make it causal evidence." The analyst's job is not to withhold judgment forever; it is to price the judgment correctly and name the purchase that would upgrade it. That framing — evidence has a price list, and experiments are how you pay — is the bridge on which this chapter ends.

CONCEPT

The Verb Is Part of the Finding

A drivers deliverable makes two kinds of claims at once: what the numbers are, and what kind of evidence they constitute. The second claim lives in the verbs, and it is as checkable as the first. Before any deliverable ships: read every sentence that contains a coefficient and grade its verb against the method that produced it; put the qualification inside the sentence the meeting will quote, because compression strips everything outside it; and when your number surfaces later wearing a stronger verb, correct it as you would correct a wrong figure. An analysis can be arithmetically perfect and still be wrong in the only sentence anyone remembers.

Source: Course concept developed for this guide, informed by Freedman (1991) and Wasserstein and Lazar (2016).

7.16 Chapter Summary

This chapter took up the question Chapter 6's treatment map could not answer — what moves these customers? — and equipped the budget-lock decision with an honest drivers analysis. The main point is the standing distinction the chapter opened with and never set down: association is what marketing data records, causation is what marketing decisions require, and the analyst's value lies in measuring the first rigorously while refusing to let it impersonate the second. "Driver" entered the chapter as a word on probation, and it leaves the same way — usable, but only with its evidence named.

The chapter's machinery arrived in two movements. The measurement movement built covariance — with the population quantity and its sample estimate distinguished, because the software reports one of them and the definition had better name which — and then the Pearson correlation coefficient, whose division by two standard deviations makes it indifferent to the convention that the covariance is not. It then cataloged what the correlation cannot buy: nonlinearity, outlier sensitivity, and mixed populations, each answered by the always-plot discipline of Anscombe's quartet, and each demonstrated on StyleCraft's base, where a hand-computed r of 0.20 between frequency and monetary value taught the mixed-population lesson on ten rows. Confounding gave the rivals of every association their mechanism, with the confounder defined by position — a pre-existing common cause — rather than by the loose test of being correlated with both, and with the calendar installed as marketing's standing confounder. Segment composition entered on different terms: an estimated segment label built from same-window behavior is a descriptive stratifier that exposes what a pooled number is averaging over, not a clean pre-treatment control, and the chapter was careful to claim only the first.

The modeling movement built regression from the fitted line upward: coefficients read aloud in native units, least squares as a criterion with a mean's sensitivities, residuals as the model's confession, and R^2 as a baseline comparison whose familiar properties hold under stated conditions — in-sample, with an intercept, on the same observations, in nested models. Multiple regression delivered the ceteris paribus reading and its strict boundaries: other included things equal, an adjustment computed from a fitted surface rather than a matching of real customers, and an estimate that can extrapolate where few customers share the relevant combination of values. Omitted variable bias supplied the direction logic for reasoning about everything excluded, with the qualification that in a multiple regression the relevant association is the one that survives the model's other predictors. Dummy variables and reference categories brought the four segments into the model as evidence, with the intercept correctly described as

the expected response at every predictor's zero rather than as the reference group's mean. The interaction term let one association differ across groups, read conditionally and reported as two dollar figures rather than one significance verdict.

The evidence-reading section calibrated the uncertainty columns at working level and connected them to the diagnostic that precedes them: a funneling residual plot is heteroskedasticity, heteroskedasticity distorts standard errors without biasing coefficients, and the repair is a robust covariance estimator requested at fit time — so every model in the labs was fitted with HC3, and the labs printed the nonrobust and robust standard errors side by side to show exactly what changes and what does not. The p-value was defined through the test statistic it is computed from, under the null hypothesis and the model's assumptions, and read with the American Statistical Association's three warnings attached. Multicollinearity was diagnosed with variance inflation factors as well as pairwise correlations, deliberately without a universal cutoff, and described honestly: it may leave in-sample fit largely intact while destabilizing coefficients, and it can weaken out-of-sample prediction when the relationships among predictors shift. Prediction and explanation were separated as modeling goals with different tolerances.

The AI section named the assistant-era failure modes — the confident causal narration, the silent specification, the plausible-but-impossible coefficient, the default covariance estimator on funneling residuals, the unconditional reading of a conditional coefficient — and installed the four-point audit of signs, magnitudes, units, and diagnostics that the labs then rehearsed. Eighteen numbered cells carried the argument into executable form: the two covariances computed side by side, the labeled miniature scatter with its mean lines, a correlation matrix preceded by the completeness check that makes it one exhibit rather than several, whale sensitivity, faceted and within-segment views of the discount claim, the composition features built and reconciled against certified totals, robust and nonrobust inference contrasted, the funnel measured rather than eyeballed, the specified model with its encoding asserted, a coefficient table carrying units, variance inflation factors, Chapter 5's decomposition remeasured, the designed interaction reported as two conditional associations, the email claim tested rather than assumed, a specification sweep, and a leakage demonstration in which a model handed the outcome under another name returns an R^2 of one. The interpretation, practice, and ethics sections carried the findings into the meeting room, where the verbs proved to be part of the finding and the analyst of record proved to be their editor.

Looking ahead, the drivers model was built to explain: its coefficients were the deliverable, and their honesty was the discipline. The next question is the one the model was not built for — how well do these patterns hold on customers the model has never seen? The moment a model's outputs, rather than its coefficients, become the product, the obligations change: targets and horizons must be defined, data must be split, baselines must be beaten, and a new class of silent failure — the model that grades itself on information it should never have had, of which Code 7.18 was a deliberately obvious instance — becomes the central danger. Chapter 8 takes up predictive modeling for marketing decisions, where Part II's machinery turns from explaining the past toward scoring the future, and where the too-good result replaces the wrong verb as the red flag an analyst is paid to notice. Further out, Chapter 11 supplies the instrument this chapter kept naming and could not use: experimental design, where random assignment severs the backdoors that every observational coefficient in this chapter had to reason around, and where the verbs this chapter rationed can finally be earned.

7.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: predict signs before computing, plot before believing, read every coefficient in its units against its reference, read conditional coefficients conditionally, reason omitted-variable-bias directions, match the covariance estimator to the residuals, run the four-point audit on every fitted model, and grade every verb against its evidence. They are organized into two groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Assignment #3, Drivers and Predictive Modeling, draws on the homework sets of this chapter and Chapter 8. In-class activities are prepared for discussion rather than submitted. Each exercise also carries its assignment label — required practice, homework submission, or in-class discussion — so that instructors can assign selectively.

7.17.1 Core Chapter Practice

Exercise 7.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

20. State the difference between association and causation, and give the opening case's app claim as an example of the gap between them.
21. Why is covariance a direction detector but not a strength measure, and what operation converts it into one? State also which covariance pandas reports, and explain why the same choice does not change the correlation.
22. An analyst reports r = 0.02 between tenure and average order value and concludes the two are unrelated. Name two distinct ways this conclusion could be wrong, citing the section that covers each.
23. Define a confounder by its position rather than by its correlations. Then explain why "related to both variables" is too broad a test, naming the two other kinds of variable that pass it and stating why adjusting for each causes a different problem.
24. State the ceteris paribus reading of a multiple regression coefficient, and then state three distinct boundaries on it — one about the variable list, one about what the adjustment actually is, and one about the regions of the data where it may be extrapolating.
25. Why must one category be excluded when a categorical variable enters a regression as dummies, and what does the intercept represent in a model that also contains a numeric predictor?
26. A model contains app_user, loyalty_enrolled, and their product. State what each of the three coefficients means, and write the expression for the app association among loyalty-enrolled customers.
27. State the three familiar properties of R^2 together with the conditions under which they hold, and give one situation in which a reported R^2 would not satisfy them.
28. A coefficient has p = 0.001 and a 95 percent confidence interval of $0.40 to $0.62 per unit. Write the honest one-sentence report, state precisely what quantity the p-value is the probability of, and name the misreading the p-value alone invites.
29. A residual plot funnels outward as fitted spend rises. Say what that is called, what it does and does not distort, and what the analyst should change about the fit. Then state what stays exactly the same after the change.
30. What does a variance inflation factor measure that a pairwise correlation matrix cannot, and why does this guide decline to publish a cutoff for it?
31. State the difference between a model built to explain and a model built to predict, and name one practice that is a failure in one and tolerable in the other.

Exercise 7.2 Predict the Signs (Required Practice)

Before running Lab 7.1 Part B, write the predicted sign (positive, negative, or near zero) of the correlation between monetary and each of: recency_days, frequency, aov, discount_share, store_share, orders_per_month, revenue_per_month, app_user, and email_opt_in — with one sentence of reasoning each, citing Chapter 5's certified findings or Chapter 6's segment profiles where they apply. Mark in advance which two of the nine you expect to be arithmetic relatives of monetary rather than behavioral findings, and say how you would tell the difference from the matrix alone. After running Code 7.4, grade every prediction and write one paragraph on your largest miss: what did the data know that your reasoning did not, or what did the pooled computation conceal that Section 7.4's tools would reveal?

Exercise 7.3 Read the Coefficient Aloud (Required Practice)

For each fitted result below, write the single honest sentence that reports it — units, direction, held-constant clause, and a verb that stops at the evidence. Then write the overclaimed version a deck would produce, and underline the exact words that changed the claim.

24. In the Code 7.11 spend model: recency_days coefficient −0.85, HC3 interval −1.00 to −0.70.

25. In the Code 7.11 spend model: seg_suburban coefficient −180, HC3 interval −220 to −140, with Urban Loyal Core as the reference.
26. In the Code 7.11 spend model: discount_share coefficient −310, HC3 interval −390 to −230. Be careful with the units on this one; state what a one-unit difference in discount_share actually is.
27. In the Code 7.14 controlled AOV model: seg_suburban coefficient 4, HC3 interval −2 to 10.
28. In the Code 7.15 interaction model: app_user coefficient 45, app_user:loyalty_enrolled coefficient 60, both intervals excluding zero. Report the two conditional associations rather than the three coefficients, and explain why the deck should never quote the 45 on its own.

Exercise 7.4 Hand Arithmetic on the Miniature (Required Practice)

Using only the ten-customer miniature of Code 7.1 and no code: verify the sample covariance of 26.73 between frequency and monetary by computing the ten deviation products and summing them, and state what the population covariance would be and why the two differ. Explain in one sentence why C002 and C010 contribute exactly zero to the sum despite being the two largest spenders. Recompute r with C006 removed and state what the change reveals about single-observation influence. Sketch the scatterplot by hand with both mean lines drawn, label the quadrant each customer falls in, and mark the region that would mislead a straight-line summary. Close with one sentence connecting the exercise to Anscombe's argument.

Exercise 7.5 The Full Drivers Deliverable (Homework Submission)

Complete Labs 7.1 and 7.2 on the certified files and assemble the VP's one-page deliverable per Section 7.13: the four verdict paragraphs on the app, discount, email, and average-order-value findings; the coefficient table with HC3 intervals and a units column; the pooled-versus-within-segment discount exhibit with its faceted plot; the two conditional app associations from the designed interaction, in dollars; the specification-sensitivity range for any coefficient your recommendation leans on; and the closing line naming the experiment that would upgrade the weakest verdict. State once, in a methods line, that all intervals are heteroskedasticity-robust, and say why. Submissions are graded on the verdict paragraphs' verbs and the audit evidence as heavily as on the code.

Exercise 7.6 AI Regression Audit (Homework Submission)

Give an AI assistant the customer_features table and this deliberately loose prompt, verbatim: "Find what drives customer spend and tell me what to do about it." Then audit the response with Table 7.5's four points. Check every reported sign against your Exercise 7.2 predictions. Check every magnitude against a hand-computed anchor — group means, Chapter 5 gaps — and note whether any adjusted figure exceeds its descriptive counterpart, which is permitted but requires an explanation grounded in the model's variables. Restate every quoted coefficient in its units, flag any cross-unit comparison, and flag any main effect quoted unconditionally from a model that contains its interaction. Determine whether the assistant examined residuals, which covariance estimator it used, and whether it connected the two. Separately, list every causal verb in the reply and grade each against Section 7.2's registry. Report what specification the assistant chose — variables, reference category, interactions — and whether it disclosed choosing. Document the full exchange per the AI-use documentation template in Appendix D, and conclude with two sentences on which audit point caught the most serious problem.

7.17.2 In-Class Activities

Exercise 7.7 Grade the Verbs (In-Class Discussion)

Each sentence below quotes a defensible analysis and states an indefensible conclusion, or states a defensible conclusion in indefensible language — or is honest as written. Classify each, rewrite the failures, and cite the section that governs your verdict.

17. "Customers acquired through influencer channels spend 30 percent more; shifting acquisition budget to influencers will raise average customer value 30 percent."

18. "Email-engaged customers buy more often, holding segment and tenure constant; email engagement is associated with higher frequency, and a frequency test would tell us whether more email causes more buying."
19. "The interaction term was significant at $p < 0.01$, confirming that loyalty drives app value."
20. "Suburban customers' AOV premium runs almost entirely through basket size and occasionwear mix." Classify this one carefully: the arithmetic behind it is correct and the verb still fails. Name what kind of claim "runs through" makes, and rewrite it as the adjustment claim the two models actually support.
21. "R^2 was 0.94, so the spend model is excellent and its coefficients can be trusted."
22. "The email association is consistent across all four segments — the interaction test came back insignificant."
23. "We cannot say anything about discounts because correlation is not causation."

Exercise 7.8 Find the Flaw in the Drivers Analysis (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

15. An analyst regresses monetary on aov, frequency, and recency, reports $R^2 = 0.96$, and recommends the model as near-perfect. Explain both why the fit is so high and why it is not exactly 1, and state what specification would make it exactly 1.
16. A deck ranks drivers by raw coefficient size: "recency (−0.85) matters less than app ownership (+140)."
17. A weekly-grain analysis finds sends and revenue correlate at 0.8 and recommends doubling December send volume.
18. A model of spend includes all four segment dummies plus an intercept, and the software drops one "for technical reasons" the analyst does not mention.
19. An analyst inspects the residual plot, notes the funnel, writes it up as a limitation, and then quotes the default confidence intervals from the same fit in the deliverable.
20. Two engagement variables are each insignificant in the full model, so the analyst concludes engagement does not matter — although the model fits far worse when both are removed, and both carry variance inflation factors above anything else in the table.
21. A colleague reports the discount–spend association is negative "even after controls," but the control list omits segment membership; state the likely bias and its direction using Section 7.8's logic, and state what the direction rule requires you to reason about that a raw cross-tabulation of segment against discount exposure would not settle.
22. An analyst adjusts for a variable measured after the exposure of interest — customers' post-campaign satisfaction scores — on the grounds that it correlates with both the campaign and spend, and reports that the campaign association disappeared.

7.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Association. The tendency of two variables to move together in observed data, so that knowing one improves a guess about the other; symmetric, observational, and consistent with multiple causal structures (adapted from Simon, 1954; Freedman, 1991).

Causation. The relation in which changing one variable, with all else undisturbed, would change another; directional and interventional, and never established by association alone (adapted from Simon, 1954; Freedman, 1991).

Ceteris paribus. The "other things equal" reading of a multiple regression coefficient: the estimated difference in the response per unit difference in a predictor among observations alike on the model's other included variables, computed as an adjustment under the model's assumed functional form rather than by matching observations (adapted from Wooldridge, 2020).

Confidence interval (coefficient). The range of coefficient values compatible with the data at a stated confidence level, centered on the estimate; its width is the finding about precision, and its validity depends on the standard error it was built from (adapted from Wasserstein & Lazar, 2016; Wooldridge, 2020).

Confounder. A pre-treatment variable that influences, or helps determine, both the predictor or exposure and the outcome, creating or distorting their observed association. Correlation with both variables is a warning sign rather than proof: a variable on the pathway between them, or a common consequence of them, is also correlated with both and requires different handling (adapted from Simon, 1954; Freedman, 1991).

Correlation matrix. The grid of pairwise Pearson correlations among a set of variables; a screening exhibit whose characteristic misreadings — ranking drivers, celebrating near-perfect cells, treating cells as levers — this chapter catalogs, and whose cells are computed pair by pair over the observations complete for that pair, so incomplete columns silently change the population behind each figure (behavior per pandas development team, 2026).

Covariance. The mean product of two variables' paired deviations from their means in a population; estimated from a sample by dividing the sum of the paired deviation products by $n - 1$. Positive for same-direction movement, negative for opposite, with a magnitude that depends on both variables' units (adapted from Rodgers & Nicewander, 1988).

Dummy variable. An indicator predictor equal to 1 for membership in a category and 0 otherwise, by which categorical variables enter regression; c categories require $c - 1$ dummies (adapted from Wooldridge, 2020).

Heteroskedasticity. Residual variance that differs systematically across the range of fitted values, visible as a funnel in a residual plot. It does not bias the fitted coefficients and it does distort the standard errors computed under the default assumption, so the intervals and p-values built from them are unreliable until a robust covariance estimator is used (adapted from Wooldridge, 2020; MacKinnon & White, 1985).

Interaction term. A predictor formed as the product of two others, whose coefficient estimates how the association between one predictor and the response differs across values of the other; once it is present, each main-effect coefficient is the association for its variable when the interacting partner equals zero (adapted from Wooldridge, 2020; James et al., 2021).

Least squares. The estimation criterion that chooses regression coefficients to minimize the sum of squared residuals. The solution is unique when the predictor matrix has full column rank; under perfect redundancy among predictors, software may still return a fitted result whose individual coefficients are not identified. Like every mean, it is sensitive to extreme observations (adapted from James et al., 2021).

Multicollinearity. Strong overlap in the information carried by a model's predictors, whether pairwise or between one predictor and a combination of others, which leaves joint explanatory power largely intact while making individual coefficients imprecise and unstable across specifications (adapted from Wooldridge, 2020).

Multiple regression. The modeling of a numeric response as a linear function of two or more predictors fitted jointly, so that each coefficient carries a ceteris paribus reading with respect to the model's other variables (adapted from Wooldridge, 2020).

Omitted variable bias. The distortion of an included predictor's coefficient when a variable related to both the response and that predictor is left out of the model. In the simplest linear case its direction follows the product of the omitted variable's effect on the response and its association with the included predictor; in a multiple regression the relevant association is the part remaining after the model's other predictors are taken into account (adapted from Wooldridge, 2020).

p-value (coefficient). The probability, assuming the stated null hypothesis and statistical model are correct, of obtaining a test statistic at least as extreme as the one observed; for a regression coefficient, the null hypothesis is that the coefficient is zero given the model's other variables. It is a measure of surprise under that hypothesis, not of size, importance, or truth (adapted from Wasserstein & Lazar, 2016).

Pearson correlation coefficient (r). The sample covariance of two variables divided by the product of their sample standard deviations: a unit-free, symmetric measure of linear association ranging from -1 to $+1$, unaffected by the denominator convention that changes the covariance (adapted from Pearson, 1895; Rodgers & Nicewander, 1988).

R^2 (coefficient of determination). The proportion of a response's variation around its mean that a fitted model accounts for. For in-sample ordinary least squares fitted with an intercept to the same observations, it lies between 0 and 1, cannot decrease when predictors are added to a nested model, and equals r^2 in the simple one-predictor case; it grades fit to the sample and never, by itself, usefulness for a decision (adapted from Wooldridge, 2020; James et al., 2021).

Reference category. The category of a categorical predictor represented when all of its dummy variables equal zero, against which every dummy coefficient is read as an adjusted difference. With an intercept in the model, the intercept is the expected response for the reference category when all other predictors are at their coded zero or reference values — the reference group's observed mean only when the model contains nothing else (adapted from Wooldridge, 2020).

Residual. An observation's actual response minus the model's predicted value; the itemized record of what the model missed, whose patterns are the primary diagnostic evidence about model form (adapted from James et al., 2021).

Robust standard error (heteroskedasticity-consistent). A standard error computed under an estimator that does not assume constant residual variance, requested at fit time and used throughout this chapter's labs in its HC3 form. It changes the standard errors, confidence intervals, and p-values, and leaves the fitted coefficients and predictions unchanged (adapted from MacKinnon & White, 1985; Seabold & Perktold, 2010).

Simple linear regression. The modeling of a numeric response as an intercept plus a slope times one predictor, with coefficients estimated by least squares and read in the native units of the variables (adapted from James et al., 2021).

Spurious correlation. An observed association that reflects only the operation of confounders or chance, so that neither variable responds to the other (adapted from Simon, 1954).

Standard error (coefficient). The estimated sampling variability of a fitted coefficient — how much the estimate would move across repeated samples from the same population — and the quantity from which confidence intervals and test statistics are built. It describes the precision of the estimate, not the spread of the underlying variable (adapted from Wooldridge, 2020).

Variance inflation factor. A diagnostic measuring how much a coefficient's estimated variance is inflated by the predictor's relationships with all the other predictors together, and therefore able to detect overlap that no pairwise correlation reveals. It is a diagnostic rather than a verdict, and this guide publishes no universal cutoff for it (adapted from Seabold & Perktold, 2010; Wooldridge, 2020).

7.19 Further Readings

Students who want additional background may begin with the following readings. Foundations of the association–causation discipline are listed first, methods second.

- Anscombe (1973) for the four-dataset argument that statistics without graphs deceive — three pages, one figure, and the origin of this chapter's always-plot discipline.
- Freedman (1991) for the classic essay on what regression can and cannot establish — "shoe leather" as the name for the evidence-gathering that no model replaces; the extended companion to Sections 7.2 and 7.15.
- Shmueli (2010) for the definitive statement of prediction and explanation as distinct modeling goals; the foundation under Section 7.10 and the doorway to Chapter 8.
- Wasserstein and Lazar (2016) for the American Statistical Association's consensus statement on p-values — short, official, and the corrective to most of what students have absorbed about significance.
- James et al. (2021), Chapter 3, for the standard accessible treatment of linear regression, one level of formality above this chapter.
- Chan and Perry (2017) for a candid practitioner account of media mix modeling's challenges — the industrial-scale version of every fragility this chapter names.

7.20 References

Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician, 27*(1), 17–21. https://doi.org/10.1080/00031305.1973.10478966

Borden, N. H. (1964). The concept of the marketing mix. *Journal of Advertising Research, 4*(2), 2–7.

Chan, D., & Perry, M. (2017). *Challenges and opportunities in media mix modeling*. Google Inc. https://research.google/pubs/challenges-and-opportunities-in-media-mix-modeling/

Freedman, D. A. (1991). Statistical models and shoe leather. *Sociological Methodology, 21*, 291–313. https://doi.org/10.2307/270939

Galton, F. (1886). Regression towards mediocrity in hereditary stature. *Journal of the Anthropological Institute of Great Britain and Ireland, 15*, 246–263. https://doi.org/10.2307/2841583

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

MacKinnon, J. G., & White, H. (1985). Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics, 29*(3), 305–325. https://doi.org/10.1016/0304-4076(85)90158-7

pandas development team. (2026). *pandas.DataFrame.corr* [Software documentation]. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London, 58*, 240–242. https://doi.org/10.1098/rspl.1895.0041

Rodgers, J. L., & Nicewander, W. A. (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician, 42*(1), 59–66. https://doi.org/10.1080/00031305.1988.10475524

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference* (pp. 92–96). https://doi.org/10.25080/Majora-92bf1922-011

Shmueli, G. (2010). To explain or to predict? *Statistical Science, 25*(3), 289–310. https://doi.org/10.1214/10-STS330

Simon, H. A. (1954). Spurious correlation: A causal interpretation. *Journal of the American Statistical Association, 49*(267), 467–479. https://doi.org/10.2307/2281124

Vigen, T. (2015). *Spurious correlations*. Hachette Books.

Wasserstein, R. L., & Lazar, N. A. (2016). The ASA's statement on p-values: Context, process, and purpose. *The American Statistician, 70*(2), 129–133. https://doi.org/10.1080/00031305.2016.1154108

Wooldridge, J. M. (2020). *Introductory econometrics: A modern approach* (7th ed.). Cengage Learning.

Chapter 08

Predictive Modeling for Marketing Decisions

From Explaining the Past to Scoring the Future

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops predictive modeling as a distinct analytic practice with its own obligations and failure modes. It builds the predictive frame — unit, target, features, and horizon — and the label as a manufactured measurement, with a snapshot date placing a wall in time between what a model may know and what it must predict. Leakage is defined as the canonical error and cataloged by route. Honest evaluation follows: train/test splits and the in-sample flattery they defeat; mean and last-value baselines operationalized as stated opponents; MAE and RMSE read in dollars beside capture at the program's capacity as the decision metric; overfitting and underfitting read from the error signature; and training-only cross-validation as the instrument that keeps the test set sealed. The labs build a spend-prediction pipeline, compare candidates on a frozen leaderboard, manufacture and catch a leaking model, and grade a vendor's near-perfect accuracy claim.

KEYWORDS

predictive modeling; leakage; train/test split; baselines; overfitting; cross-validation; MAE; RMSE; expected value; model drift

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Predictive modeling for marketing decisions. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 8, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, NumPy, Matplotlib, and scikit-learn are sponsored or affiliated projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. statsmodels is a community-developed project distributed under the modified BSD license. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the

underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

37. Distinguish predictive from explanatory modeling as goals with different obligations, citing the Section 7.10 distinction, and state what changes when a model's outputs become the product.
38. Specify a complete predictive frame — unit of prediction, target, feature set, and horizon — in writing before fitting, as the predictive extension of the analytic specification of Section 2.5, and declare its decision metric and baselines alongside it.
39. Define a prediction label precisely, including the snapshot date, feature window, outcome window, and eligibility rule, and explain why the label is a measurement decision rather than a given.
40. Define leakage, catalog its common routes in marketing data, and audit a feature list for information that crosses the snapshot line.
41. Construct an honest train/test evaluation, explain in-sample flattery, state what a random customer holdout does and does not validate, and apply the split hygiene rules that protect it.
42. Compute and interpret MAE and RMSE in dollars, compute top-decile spend capture as the decision metric of a ranked-list program, and choose the metric whose weighting matches the decision at stake.
43. Operationalize baselines — mean, majority-class, and last-value — as stated opponents, and express a model's value as its margin over the strongest honest baseline.
44. Diagnose overfitting and underfitting from the training-versus-validation error signature, and explain the bias–variance tradeoff at intuition level.
45. Run k-fold cross-validation on the training data to compare candidate models and select model complexity, and apply the comparison discipline of same folds, same metric, same baselines.
46. Translate predictions into decisions: build ranked targeting lists under a fixed budget, apply expected-value framing at concept level, distinguish predicted spend from spend caused by a treatment, and state the deployment realities of data drift, concept drift, performance decay, and retraining.
47. Verify an AI-built or vendor-supplied predictive model with the five-point audit — frame, leakage, split hygiene, baseline, and error in decision units — treating “too good” as a red flag, and document the audit per Appendix D.

Chapter 7 ended with a deliverable and a warning. The deliverable was the drivers analysis: spend and average order value modeled on the feature table, segment dummies read against the Urban Loyal Core, the designed interaction priced in dollars, and every coefficient guarded by a verb that stopped at the evidence. The warning came in the chapter's last paragraph: the drivers model was built to explain, and the moment a model's outputs — rather than its coefficients — become the product, the obligations change. This chapter is about that change. The question is no longer what moved spend in the window we observed; it is what spend will be in the window we have not yet seen, customer by customer, accurately enough to act on. Answering it requires machinery Chapter 7 never needed: a target defined in time rather than assumed, a dataset split so the model can be graded on customers it never met, baselines that convert Chapter 2's interpretive discipline into a working opponent, error metrics denominated in dollars and in the decision's own currency, and a new class of silent failure — leakage, the model that grades itself on information it should never have had — whose most reliable symptom is the one result every stakeholder wants to hear: a model that looks too good.

CONCEPT

What This Chapter Is Really About

Chapter 7 closed by separating two modeling goals: explanation, where the coefficients are the deliverable, and prediction, where the outputs are. The separation sounds administrative and is anything but, because the two goals fail differently. An explanatory model fails in public — a sign that contradicts known structure, a causal verb that overreaches — and Chapter 7 built the audit that catches it. A predictive model fails in private. It can be arithmetically flawless, statistically significant, beautifully fitted, and worthless, because everything that certified it happened on data it will never see again.

The predictive disciplines of this chapter — the frame declared before fitting, the wall in time between features and outcome, the test set touched once, the opponent named before the race — are all versions of a single idea: the model must be graded under the conditions of its actual job, predicting what it has not seen, against the alternative of not using it at all. And the chapter's characteristic red flag inverts every instinct a student brings to it: in predictive work, the result that looks too good almost never is, and the analyst's most valuable reflex is suspicion of their own success.

Source: Course concept developed for this guide, informed by Shmueli (2010) and Kaufman et al. (2012).

8.1 Marketing Decision Context: The Scored List and the Vendor's Promise

Chapter 7's drivers deliverable survived the budget-lock meeting: the install campaign was funded as a test rather than a certainty, the discount ladder survived on the within-segment exhibit, and the fall retention program's incremental budget was allocated across the four journeys the treatment map defined. Now the program is being built, and its most expensive component has a capacity constraint. The top journey tier — internally, the Backstage Preview — pairs early access to the holiday drops with a printed lookbook and a small gifting touch. Merchandising and print work out to roughly twenty dollars per treated customer, and the budget covers eight hundred customers: the top decile of a base of roughly eight thousand. Eight hundred slots, eight thousand customers. Someone must decide who gets one, and the CRM manager has asked the obvious question in its modern form: can we predict, customer by customer, who will spend the most over the next six months — and use that prediction to fill the list?

Three answers are already in the building, and each conceals a problem this chapter exists to expose. The first is the incumbent rule, defended by the CRM team's operations lead: “Rank customers by what they spent in the last six months and take the top eight hundred. Past spend is the best predictor of future spend. We do not need a model; we need a spreadsheet sorted in descending order.” The second arrives from outside, in a vendor deck. StyleCraft's customer-data-platform vendor has been piloting a predictive-spend module, and its pitch slide is the kind that ends meetings: “Our AI model predicts each customer's next-six-month spend with 96 percent accuracy ($R^2 = 0.96$).” The annual license costs more than the entire Backstage print budget. The third comes from inside the analytics team, and it is the most tempting because it is nearly right: “We already built a spend model in the drivers analysis. Its signs survived the four-point audit — score everyone with it and take the top eight hundred.”

Each answer has a real virtue and a disqualifying flaw, and stating the flaws is the fastest tour of this chapter's territory. The incumbent rule is cheap, transparent, and — this chapter will insist — the legitimate baseline every model must beat before it earns a budget line. But Chapter 5's certified findings already show why it misfires: the base is lumpy. Suburban Occasion customers concentrate a season's spending into one or two large baskets, so a customer with zero recent spend may be three weeks from a four-hundred-dollar occasionwear order, and a customer who just placed one may be done for the year. A backward-looking sort systematically gifts the customers whose spending just happened and misses the customers whose spending is about to. The vendor claim fails differently: not on its arithmetic, which no one at StyleCraft can inspect, but on its vocabulary and its plausibility. Chapter 7

established that individual human spending is noisy and that fit statistics near 1 on individual behavior are a symptom rather than an achievement (Section 7.7). The in-house answer fails most instructively of all. The drivers model was built, in Section 7.10's terms, to explain: fitted on the full window, graded on the data it was fitted to, its variables chosen to make coefficients defensible rather than predictions accurate. Nothing about surviving the four-point audit certifies that its predictions hold on customers and months it has never seen.

One boundary belongs in the brief rather than in a footnote three sections later, because it constrains what the whole project can claim. A model of future spend can prioritize the customers expected to be most valuable. It cannot identify the customers whose spending will increase *because* they receive the Backstage treatment. Those are different quantities, and only the second is what a marketer means by the return on a gift. The program is therefore commissioned as a recognition and prioritization program — treat the customers we expect to matter most — with the incremental question named as unanswered and Chapter 11's experimental machinery named as the instrument that would answer it.

So the VP of Marketing has commissioned the analyst — again, you — with a deliverable and a deadline. The deliverable: a scored list of eight hundred customers, produced by a method chosen honestly among the three candidates or a better fourth, with the evidence for the choice stated in a form the CFO can interrogate. The deadline: the scoring method must be locked in thirty days, when the holiday-drop calendar goes to print. Sections 8.2 through 8.4 build the frame, the label, and the leakage discipline. Sections 8.5 through 8.9 build honest evaluation: splits, error metrics, baselines, overfitting, and the comparison rules. Section 8.10 carries the score to the decision and into deployment. Section 8.11 turns to the AI assistant, which will build any pipeline fluently — including, without complaint, a leaky one. The labs in Section 8.12 build the honest pipeline, compare candidates on a frozen leaderboard, manufacture the leaky one on purpose, and catch it. Sections 8.13 through 8.15 rehearse the meeting.

8.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

48. The incumbent rule — rank by trailing six-month spend — is both a candidate answer and, in this chapter's terms, a baseline. What is the difference between rejecting it as an answer and using it as a baseline, and why does the second remain essential even if the first happens?
49. The vendor's "96 percent accuracy" claim contains two separate problems: one of vocabulary and one of plausibility. Name both, and state which sections of this guide equip you to press on each.
50. The in-house drivers model survived Chapter 7's four-point audit. List two specific reasons that audit does not certify the model for the scoring job, drawing on the distinction in Section 7.10.
51. The brief separates predicting who will spend most from identifying whose spending the gift would increase. State the practical consequence of that boundary for how the program's success should be measured next quarter.
52. The Backstage list will treat eight hundred customers and pass over seventy-two hundred. Describe one way each kind of error — gifting a customer who was going to spend little, missing a customer who would have spent much — costs the program, and recall which earlier section taught you to expect the two costs to differ (the answer returns in Section 8.10 and again in Chapter 9).

8.2 From Explanation to Prediction: The Predictive Frame

Section 7.10 drew the distinction this chapter now inhabits: explanation and prediction are different modeling goals, served differently by the same equation, with different success measures and different characteristic failures. One sentence of reminder suffices — an explanatory model is graded on whether its coefficients are honest structure; a predictive model is graded on whether its outputs are accurate for cases it has not seen — and the rest of this section develops what the second goal demands before any model is fitted at all. The demand is a specification, and it is the direct descendant of the analytic specification of Section 2.5: just as no analysis in this guide begins without a written statement of the decision, the alternatives, and the deliverable, no predictive model begins without a written statement of four commitments this guide calls the predictive frame.

DEFINITION

Predictive Modeling and the Predictive Frame

Predictive modeling is the construction of a model whose purpose is to output accurate estimates of an unknown or future quantity for individual cases, evaluated on cases not used to build it. Its specification — the predictive frame — consists of four declared commitments: the unit of prediction (what entity receives a prediction), the target (the quantity predicted, precisely defined), the features (the information the model may use, fixed as of a stated moment), and the horizon (how far beyond that moment the target lies).

Source: Adapted from Provost and Fawcett (2013) and Kuhn and Johnson (2013).

Each commitment is a decision with consequences, and each connects to machinery this guide has already built. The unit of prediction is the grain question of Section 3.3 pointed forward: StyleCraft's scoring problem lives at the customer grain — one prediction per customer — but the same business could ask for predictions at the store grain (next quarter's revenue per store), the campaign grain (expected response per send), or the day grain (Chapter 10's subject). Declaring the unit settles what a row means in every table the pipeline touches, and undeclared units produce the same silent damage here that undeclared grain produced in Chapter 4's joins. The target is the subject of Section 8.3, because it deserves its own section: “future spend” is not a column that exists but a measurement that must be constructed, and every construction choice changes what the model learns to do. The features are the model's permitted information, and the discipline they require — nothing the model would not know at prediction time — is the subject of Section 8.4. The word itself was defined in Section 3.5 among the variable roles; what is new here is not the role but the restriction. The horizon is the forward distance of the claim: the Backstage decision needs six months, because the program will be judged on the fall-and-holiday arc, while a win-back trigger might need only thirty days. The horizon disciplines everything downstream — a model with a six-month horizon cannot be validated in two weeks, and Section 8.10 will show that the horizon also sets the clock on how often the model must be retrained.

Table 8.1 states the frame for the Backstage decision in full, and its final column carries the point of the exercise: every row is a choice that could have been made differently, and each alternative would produce a different model answering a different question. In other words, “predict customer spend” is not yet a modeling problem — it is a family of modeling problems, and the frame is the act of choosing one member and committing to it in writing. The commitment is what makes verification possible later: a model can only be audited against a frame that exists.

Table 8.1

The predictive frame for the Backstage scoring decision

Commitment	Declaration for this decision	What a different choice would change
Unit of prediction	One prediction per customer (customer grain, per Section 3.3)	Store- or day-grain models answer planning questions, not list-building ones
Target	Total spend, in dollars, in the six months after the snapshot date (defined fully in Section 8.3)	A yes/no target (“will spend at all”) changes the method family — Chapter 9's subject
Features	The customer feature table, computed only from transactions inside the declared feature window	Features from after the snapshot are leakage (Section 8.4), not information
Horizon	Six months — the fall-and-holiday arc the program will be judged on	A 30-day horizon serves triggers, not seasonal list-building; longer horizons decay differently (Section 8.10)
Decision metric	Spend capture at the program's capacity share — 800 slots divided by the eligible population — against the incumbent list (Section 8.6)	A larger budget or a wider eligible population moves the cutoff and therefore the metric

Commitment	Declaration for this decision	What a different choice would change
Calibration metrics	MAE leading, RMSE reported beside it, both in dollars	A decision with convex costs would promote RMSE to the lead (Section 8.6)
Baselines	Mean prediction and the trailing-six-month ranking, on the same folds and the same test set	A baseline chosen after the results can be chosen to lose (Section 8.7)

One more inheritance completes the frame's paperwork, and Table 8.1's last three rows record it. Section 2.5's specification discipline required that the deliverable be named before the analysis begins; the predictive version has a sharper edge, because the frame must also name, in advance, the metric the model will be graded on and the baselines it must beat. Declaring them before fitting is not ceremony. It is the predictive analog of predict-then-verify (Section 1.7): a model builder who chooses the success metric after seeing the results has the same conflict of interest as an analyst who writes the hypothesis after seeing the data, and Section 8.9's comparison discipline exists because the temptation is real, routine, and mostly unconscious.

8.3 Defining the Target: Labels, Snapshots, and Windows

The frame's second commitment deserves its own section because it is the one students most reliably treat as given when it is in fact constructed. The drivers analysis of Chapter 7 modeled monetary — total spend across the observed window — a column that existed in the feature table before any model was imagined. The scoring decision has no such column. “What this customer will spend in the next six months” exists nowhere in StyleCraft's data today; for the model to learn it, the analyst must manufacture historical examples of it, and the manufacturing is a measurement act with all of Chapter 3's obligations attached. The manufactured quantity is called the label: the recorded value of the target for a case whose outcome is already known, from which the model learns the relationship between features and future.

DEFINITION

Label (Target Variable)

A label is the constructed, recorded value of the target for a historical case — the answer the model trains on. Constructing labels requires choosing a snapshot date that splits time into a feature window (the past the model may know) and an outcome window (the future the label summarizes), together with eligibility rules for which cases receive labels at all. The label is a measurement decision: its windows, its filters, and its eligibility rules are choices that change what the model learns, and they are declared, not discovered.

Source: Adapted from Provost and Fawcett (2013) and Kuhn and Johnson (2013).

The construction is easiest to see drawn on StyleCraft's own calendar. The certified transaction history runs twenty-four months, from July 1, 2024 through June 30, 2026 — the window every lab since Chapter 4 has used. To manufacture labels, the analyst plants a snapshot date inside that history: December 31, 2025. The feature window is the eighteen months from July 1, 2024 through December 31, 2025, and the customer feature table is rebuilt from it, exactly per Table 6.5's derivation rules but with the analysis date moved back to the snapshot. The outcome window is the six months from January 1 through June 30, 2026, and each customer's label, call it spend_next6m, is their total certified spend inside it: numerator, the sum of line_revenue on their transactions in the outcome window; denominator, none — it is a total, not a rate; window, January 1 through June 30, 2026; filters, none beyond the eligibility rule below. The four-element discipline of Section 3.5 applies to labels exactly as it applied to metrics, and writing the label in that form is not an homage — it is what makes two analysts' pipelines produce the same target.

Notice that both windows have two ends, and that both ends are load-bearing. A feature window declared as eighteen months but implemented as “everything before the snapshot” silently includes whatever history the file

happens to carry, so two analysts working from the same declaration produce different recency and different monetary values. An outcome window declared as six months but implemented as "everything after the snapshot" silently lengthens the horizon for any customer whose file extends further, so the label stops meaning what the frame says it means. Lab 8.1 therefore names four dates rather than one and asserts that the minimum and maximum transaction dates on each side fall inside their declared window. This is the four-element metric discipline turned into two assertions.

Three consequences of the construction deserve a paragraph each, because each becomes a lab step. First, the snapshot creates a wall in time, and the wall is what the whole chapter protects. On the feature side, the model may know everything StyleCraft knew on December 31, 2025: eighteen months of recency, frequency, spend, discount behavior, channel mix, and engagement. On the outcome side, the model may know nothing — not one transaction, not one field updated after the snapshot — because at the moment the real scoring decision is made, the future is exactly that unavailable. Every feature must therefore be re-derived as of the snapshot: recency_days recomputed against December 31, tenure_days truncated at the wall, monetary summing only feature-window transactions. Reusing the June 2026 feature table, the one Chapters 6 and 7 worked with, would hand the model six months of the very future it is being asked to predict — the error Section 8.4 names.

Second, eligibility is part of the label, and this chapter states its rule in the form the code can enforce. A customer receives a labeled row when two conditions hold: their signup_date falls on or before the snapshot, and they placed at least one order inside the feature window. The first condition is obvious — a customer acquired in March 2026 has no feature-window history and would enter the training data as a row of zeros wearing a label they never had a chance to earn. The second is the one drafts forget, and it is not a convenience: recency, average order value, discount share, and store share are all undefined for a customer with no feature-window purchases, because each of them divides by a quantity that is zero. This is the same population rule Chapter 6 declared for the clustering base (Section 6.6), inherited here deliberately rather than reinvented, and it carries the same obligation: the excluded customers are counted and named, not silently dropped.

Third, a label of zero is a label, not a gap. Customers eligible at the snapshot who bought nothing in the outcome window have spend_next6m equal to zero — and they are the single most important group in the training data, because they are the customers the incumbent sort-by-past-spend rule cannot distinguish from the lumpy occasion shoppers about to place a large order. Dropping zero-label rows, a mistake AI assistants make with some regularity when asked to "clean" a training table, deletes precisely the contrast the model exists to learn. The missing-data disciplines of Chapter 4 apply, but the prior question is definitional: here, absence of transactions is the outcome, not a defect in recording it.

CONCEPT

The Modeling Population Is Not the Deployment Population

The eligibility rule defines who the model learns from. It also defines, by inheritance, who the model can honestly score — and the two populations drift apart the moment the model is deployed. StyleCraft's Backstage model learns from customers who had purchased by December 31, 2025. By the June scoring run, the base contains customers acquired since the snapshot, customers who signed up long ago and bought for the first time in the spring, and customers whose only history sits outside any window the model has seen.

The instinct to say the model "cannot" score them is wrong, and the correction matters. Technically it can: the same features can be constructed for anyone with a purchase history, and the model will return a number for every row it is handed. The real constraint is representation. A customer whose feature combination barely appears in the training data receives a prediction the evaluation never tested, and a customer with no feature-window history at all receives one built from undefined ratios. Those customers are an activation problem, not a scoring problem, and the deliverable says so in writing: the model prioritizes established customers, a separate rule handles the newly acquired, and the boundary between them is declared rather than discovered in production.

Source: Course concept developed for this guide, informed by Kuhn and Johnson (2013).

8.4 Leakage: The Canonical Error

Chapter 7 closed its multicollinearity section with a designed trap: regress monetary on aov and frequency, watch R^2 leap toward 1, and recognize that the model had been handed the answer — monetary is, by Table 6.5's own derivation rules, their product. Chapter 7 promised that Chapter 8 would name the general disease. Here is the name. Leakage is the use, in training or evaluating a predictive model, of information that would not be available at the moment the model makes its real predictions — information from the future, from the target's own construction, or from the evaluation data itself. It is the canonical error of predictive modeling: the most common serious failure, the hardest to see in a finished pipeline, and the one whose symptom is a model that performs implausibly well right up until it is deployed, at which point the borrowed information is no longer there to borrow and performance collapses to what the legitimate features always supported (Kaufman et al., 2012).

DEFINITION

Leakage

Leakage is the presence, in a model's training features or evaluation procedure, of information that will not legitimately be available when the model is used for its real predictions — most often information from after the prediction moment, information arithmetically derived from the target, or information shared between training and test data. A leaky model's measured performance reflects the leaked information rather than genuine predictive relationships, and its deployed performance falls to the level the legitimate features support.

Source: Adapted from Kaufman et al. (2012).

In other words: a leaky model is not a bad model that can be tuned into a good one. It is an honest measurement of the wrong thing — of how well the future predicts itself — and no amount of refitting repairs it, because the defect is in the data the model was allowed to see. That is why leakage is caught by audit rather than by statistics: no diagnostic printed by the fitting software distinguishes leaked performance from earned performance, and the only reliable detector is an analyst tracing where each feature came from and when it became knowable. The trace has a natural checklist, because leakage in marketing data arrives by a small number of well-worn routes. Table 8.2 catalogs the five this guide's students will actually meet, each with its StyleCraft instance and its repair; the first two account for most real cases.

Table 8.2

Common leakage routes in marketing data

Route	Mechanism	StyleCraft instance	Repair
Temporal leakage	Features computed from data after the snapshot date	Building the training table from the June 2026 feature table when the label starts January 2026	Re-derive every feature as of the snapshot; audit each column's window
Target-derived (arithmetic) leakage	A feature that is an arithmetic relative of the label	Full-window monetary as a feature: it equals feature-window spend plus the label itself	Trace every feature's derivation rule (Table 6.5) against the label's
Outcome-window contamination	A feature that quietly summarizes part of the outcome window	Days since last purchase computed at the June 2026 analysis date — small values are only possible because of outcome-window orders	Recompute against the snapshot; range-check features for impossible values

Route	Mechanism	StyleCraft instance	Repair
Split contamination	A learned preprocessing or feature-construction step fitted before splitting	Standardizing features, or refitting Chapter 6's segmentation, using values that include the test customers	Fit every learned step inside a pipeline, on training data only, then apply to validation and test
Population leakage	Eligibility or filtering rules that use outcome-window information	Training only on “active customers,” with activity defined over the full 24 months	Define eligibility using feature-window information only

Three of the table's rows repay a closer look, because their surface innocence is the lesson. Outcome-window contamination is temporal leakage's subtle cousin: no one typed “use the future” — someone reused a convenient existing table whose columns were computed at the wrong date, and every derived value silently absorbed six months of the answer. The tell is often a range check, which is why the audit in Section 8.11 includes one: at a December 31, 2025 snapshot, no eligible customer can have a recency of fewer than 181 days when recency is computed correctly, so a table full of smaller values has been computed at a later date, and a single impossible value convicts the whole column.

Split contamination is smaller in effect than the first two rows and larger in principle, and it needs its boundary drawn precisely, because the boundary is where students most often over-apply the rule. What contaminates a split is a step that *learns something from the data* and is fitted before the split: a standardizer that estimates means and spreads, an imputer that estimates fill values, a feature selector that ranks columns by their relationship to the target, a clustering model that estimates centroids, a target encoder that averages the label within categories. Each of those carries a whisper of the test customers' distribution into training, and the whispers compound across steps. A transformation that learns nothing — taking a logarithm, multiplying two columns, expanding a predictor into fixed polynomial powers — does not contaminate anything, because applying it before or after the split produces identical values for every row. The rule to memorize is therefore about estimation rather than about ordering: anything estimated from data — a mean, a scale, an imputation value, a centroid, a category encoding — is estimated from the training data only, which in practice means it lives inside a pipeline that is refitted in every fold (scikit-learn developers, 2026a).

Population leakage is the one that hides inside a sentence rather than inside a column. A filter is a modeling decision, per Section 3.5, and a filter that consults the outcome window chooses the training population using the answer. “Customers who were active during the study” sounds like housekeeping and is a prediction: it quietly removes the customers who went dark, which is the outcome the model is supposed to be able to anticipate. The repair is mechanical — every eligibility condition must be evaluable using only information available on or before the snapshot — and this chapter's eligibility rule was written in Section 8.3 to satisfy exactly that test.

What makes leakage the chapter's central discipline rather than one hazard among several is its relationship to the chapter's red flag. Every route in Table 8.2 improves measured performance; none improves real performance; therefore unexplained excellence is evidence of a leak. The logic deserves a box, because it inverts the instinct every student brings to model building — and because it is the exact instrument the vendor claim of Section 8.1 requires.

CONCEPT

Too Good Is a Finding

In descriptive work, a strong result invites celebration and a write-up. In predictive work, a strong result invites an audit, and the stronger the result, the more urgent the audit. The reasoning is blunt in practice: genuine predictability of individual human behavior is bounded — Chapter 7 established that noisy individual outcomes cap what any honest model can explain — while leakage routes are numerous, quiet, and always performance-enhancing. A reported R^2 of 0.96 on individual customer spend is therefore far more likely to be a leak than a breakthrough, and the response it deserves is not admiration but the five-point audit of Section 8.11, starting with the feature trace.

The discipline extends beyond your own models. Applied to the vendor's deck, “too good” is not an accusation but a question the vendor must be able to answer: show us the frame, the snapshot, and the feature windows. A vendor who cannot is quoting in-sample fit or leaked features; either way, the number does not mean what the slide implies. This box is the chapter's verification theme in one sentence: in predictive modeling, suspicion scales with success.

Source: Course concept developed for this guide, informed by Kaufman et al. (2012) and Domingos (2012).

8.5 The Out-of-Sample Standard: Train/Test Splits and In-Sample Flattery

Leakage corrupts what the model knows; this section addresses how the model is graded, and the two disciplines are halves of one standard. A predictive model's job is to be accurate on cases it has not seen. Its performance on the cases it was fitted to — in-sample performance — is therefore not merely a weaker version of the evidence; it is systematically misleading evidence, and misleading in a known direction. Least squares, by construction, chooses the coefficients that minimize error on the fitting data (Section 7.6). The fitted model is thus tailored to that data's every quirk: its whales, its noise, its accidental patterns. Grade it on the same data and the tailoring is graded as skill. This guide calls the phenomenon in-sample flattery, and its size is not constant — it grows with model flexibility, shrinks with sample size, and vanishes only in the one place it cannot be observed: the future. Every impressive statistic computed on training data should be read with this discount in mind, and the discount cannot be computed from the training data itself.

The repair is structural, not statistical: hold data back. Before any fitting, the labeled customers are split at random into a training set, which the model may learn from without restriction, and a test set — also called a holdout — which is sealed until evaluation and consulted exactly once, for the finalist. The split is performed at the unit of prediction — whole customers go to one side or the other, per the frame's grain declaration — with a fixed random seed so the split is reproducible, and with proportions this guide sets at 80/20: enough training data to fit stable models on eight thousand customers, enough test data to grade them with useful precision.

DEFINITION

Train/Test Split and the Test Set

A train/test split randomly partitions labeled cases, at the unit of prediction, into a training set used to fit and select models and a test set held out to evaluate the selected model. Test-set performance estimates out-of-sample performance — how the model will do on cases it has not seen — which is the only performance that matters for a predictive model's real job. The estimate is honest only if the test set contributes nothing to fitting, preprocessing, or model selection, and is consulted once for the final model rather than repeatedly during development.

Source: Adapted from James et al. (2021) and scikit-learn developers (2026b).

The definition's last clause carries the discipline most often violated in practice, so it earns its own paragraph. A test set consulted repeatedly stops being a test set. If the analyst fits a model, checks test error, adjusts, checks again, and repeats, the adjustments are being steered by the test data — a slow-motion version of training on it, and a cousin of Table 8.2's split contamination. By the tenth consultation, the “held-out” error is flattery with extra steps. The working protocol, which the labs of Section 8.12 follow to the letter: development decisions — which features, which model family, how much flexibility — are made using the training data alone, with Section 8.9's cross-validation supplying out-of-sample estimates during development; the test set is opened at the end, once, to grade the finalist; and if development resumes after that, the honest analyst treats the old test error as spent. On StyleCraft's eight thousand customers this protocol costs little and buys the deliverable its central sentence: “on sixteen hundred customers the model never saw, the average error was X dollars” — a sentence the CFO can trust precisely because of everything the protocol forbade.

One qualification completes the section, and it is more important than it looks, because the chapter later teaches drift. It is tempting to say that the test customers "stand in for the future." They do not, quite. A random customer holdout estimates cross-sectional generalization at the historical snapshot: how the model performs on other customers drawn from the same population, at the same moment in the business's life, under the same acquisition mix, pricing, and merchandising. It does not measure time-forward transport — whether a relationship estimated at a December 2025 snapshot still holds at a June 2026 scoring date after the world has moved. Measuring that requires a different design: fit on an earlier snapshot, evaluate on a later one, or, once deployed, monitor realized error as each fresh outcome window matures. Standard random folds assume a stable sampling distribution, and where the intended generalization is explicitly into later periods, a time-ordered evaluation is the honest instrument (scikit-learn developers, 2026b).

The random split is nonetheless the right design here, and the reason is worth stating so the qualification is not mistaken for a defect. The frame already handled time: the wall of Section 8.3 lives inside the label construction — every customer's features stop at the snapshot and every label starts after it — so shuffling customers cannot shuffle the future into the past. What the shuffle cannot do is manufacture a second point in time, which is why the deliverable's performance sentence names its population ("customers like these, at this snapshot") and the monitoring plan of Section 8.10 exists to cover the rest. When the unit of prediction is itself a time period, as in Chapter 10's daily forecasting, random splitting breaks even the first guarantee, and a time-aware holdout is mandatory rather than supplementary.

> **BRIDGE**
>
> Chapter 10 will show why forecasting cannot randomly split its data — and what time-aware holdouts do instead.

8.6 Measuring Error in Dollars and in Decisions: MAE, RMSE, and Capture

An honest test set answers where to measure; this section answers what to measure, and the answer has two parts, because the Backstage program asks two different questions of the same predictions. The first question is calibration: how close is a customer's predicted dollar amount to what they actually spend? The second is the decision: which eight hundred customers appear above the cutoff? Those are not the same question, a model can improve on one while worsening on the other, and a frame that declares only the first has quietly agreed to be graded on something other than its job.

Take calibration first, because it supplies the vocabulary. The raw material is the per-customer error — actual label minus predicted value, the out-of-sample sibling of Section 7.6's residual — and the reporting need is a single number that summarizes sixteen hundred such errors in units a decision can use. Two summaries dominate numeric prediction, both built by this guide's oldest recipe (aggregate the individual deviations, per Chapter 5), and their difference is a policy choice about big misses. The mean absolute error takes each customer's error, strips its sign, and averages. Its virtues are bluntness and units: an MAE of 54 means the model's predictions miss actual six-month spend by 54 dollars on average, a sentence any stakeholder can absorb, denominated in the same dollars as the budget. The root mean squared error squares each error before averaging, then takes the square root to return to dollars. The squaring is the policy: it weights a single 300-dollar miss like nine separate 100-dollar misses, so RMSE is dragged upward by the largest errors far more than MAE is. Neither metric is more correct; they encode different sensitivities, and the gap between them is itself diagnostic — RMSE materially above MAE on the same predictions announces that the error is concentrated in a few large misses rather than spread evenly (Willmott & Matsuura, 2005), which on StyleCraft's data means one thing before anyone plots it: the whales and the lumpy occasion baskets of Chapter 5's concentration analysis are where the model hurts.

DEFINITION

MAE and RMSE

Mean absolute error (MAE) is the average of the absolute differences between actual and predicted values: the typical miss, in the target's native units, with every error weighted equally. Root mean squared error (RMSE) is the square root of the average squared difference: also in native units, but disproportionately sensitive to large errors, so that it exceeds MAE whenever misses are uneven and exceeds it greatly when error is concentrated in a few cases. Both are computed out of sample; the choice between them is a statement about which misses the decision cares about most.

Source: Adapted from Willmott and Matsuura (2005) and James et al. (2021).

Now the decision question, and the correction it forces on a tempting argument. It is easy to justify leading with MAE by saying that a wasted twenty-dollar slot costs the same whether the prediction behind it was moderately or catastrophically wrong, so the program's economics do not square the pain. The premise is true and the inference does not follow, because the cost of the decision is not a function of the point-prediction error at all. What costs money is whether an error moved a customer across the selection threshold, who was displaced from the list when it did, and how far the ranking shifted. A catastrophic prediction error four hundred ranks below the cutoff costs exactly nothing; a small error at rank 800 changes who receives a lookbook. Error metrics summarize distance; the program acts on order.

The metric that grades order under a fixed capacity is capture. Score every customer, sort descending, take as many customers as the budget funds, and compute what share of the population's actual outcome-window spend those customers turned out to account for. A model whose funded customers capture 34 percent of next-window spend is ordering customers better than one whose selection captures 28 percent, and the difference converts directly into dollars by multiplying the percentage-point gap by total outcome spend. Because the incumbent rule produces its own ranked list, the two can be graded head to head on exactly the artifact the meeting is deciding between, which makes the incremental capture over the incumbent the single most persuasive number the project produces.

One quantity inside that paragraph has to be settled rather than assumed, because it decides what the metric means: how many customers is “as many as the budget funds,” as a share of the population being ranked? The Backstage budget funds eight hundred treatments. That is a capacity, not a proportion — it does not grow if the eligible population turns out to be larger, and it does not shrink if the eligibility rule of Section 8.3 excludes more customers than anyone expected. The selection share is therefore computed rather than typed: eight hundred divided by the population the model can honestly score. On StyleCraft it lands near a tenth, which is why this guide is comfortable calling the exhibit top-decile capture in conversation and why Chapter 9 generalizes it to deciles. The labs still compute it, because a proportion typed into a metric is a proportion nobody re-derives when the population changes.

The frame of Table 8.1 therefore declares three metrics with three different jobs, and the labs report all three for every candidate. Capture at the capacity share is the decision metric, because the program acts through a fixed ranked cutoff. MAE is the lead calibration metric, because it gives an interpretable typical miss in the currency of the budget and keeps the deliverable's language honest about how much the model does not know. RMSE is reported beside MAE as the concentration alarm. A different decision could reorder them: if predictions fed inventory commitments, where one enormous miss costs more than many small ones combined, RMSE's sensitivity is the point, and if every customer received a personalized offer amount, calibration would become the decision metric rather than its companion. The general rule follows the asymmetric-cost discipline of Section 2.7 — choose the metric whose weighting matches the decision's actual cost structure — and note, for honesty, that percentage-based error metrics are deliberately absent here; they have a characteristic failure on near-zero actuals, and this guide takes them up where they natively live, in Chapter 10's forecasting.

CONCEPT

Grade the Model on the Artifact the Meeting Is Choosing

Every predictive deliverable in this guide reports the metric that matches the act. When the act is a ranked list under a capacity constraint, the model is graded on the list: what share of the outcome the funded customers capture, against what the incumbent's funded customers capture at the same capacity share, with the overlap counted so the meeting can see how many of the eight hundred names actually change. When the act is a dollar commitment per customer, the model is graded on calibration. Reporting only a distance metric for a ranking decision is not wrong arithmetic; it is a quiet substitution of a question the analyst can answer for the question the business asked. Chapter 9 develops the full family — lift, gains, capture by decile — where ranked targeting is the chapter-long subject; here the top-decile version carries the decision.

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013).

8.7 Baselines, Operationalized

Section 2.7 introduced baselines as interpretive discipline: no number means anything until you know what it is being compared against. Section 7.7 revealed R^2 as secretly a baseline comparison — the model against guessing the mean. This section completes the thread's promised operationalization: in predictive modeling, the baseline stops being a lens and becomes an opponent — a fully specified, deliberately simple prediction rule, evaluated on the same folds and the same test customers with the same metrics as the model, whose defeat is the model's entry fee. The reframing sounds small and changes everything about how results are reported: a model with a test MAE of 54 dollars is not good or bad in itself; it is good if the best naive rule scores 80 and embarrassing if the best naive rule scores 55.

DEFINITION

Baseline Model

A baseline model is a deliberately simple prediction rule — requiring no fitting beyond, at most, one summary statistic — evaluated under exactly the conditions of the candidate model: same folds, same test set, same metrics. Standard baselines include the mean baseline (predict the training mean for everyone; numeric targets), the majority-class baseline (predict the most common class for everyone; categorical targets), and the last-value baseline (predict that the last observed value, or window, repeats). A predictive model's demonstrated value is its margin over the strongest applicable baseline, not its raw score.

Source: Adapted from Provost and Fawcett (2013).

Table 8.3 gives the three standard baselines their StyleCraft form. The mean baseline is the floor: predicting the training-set average of spend_next6m for every customer uses no customer information at all, and a model that cannot beat it has learned nothing — it is R^2's comparison (Section 7.7), now made to stand in the open and fight on held-out customers. It is also the baseline that needs one qualification on the decision metric, and the qualification is more instructive than it looks. A rule that gives every customer the same score produces no ranking at all, so its capture is not a property of the rule; it is a property of whatever order the rows happened to arrive in, and reported carelessly that is row order masquerading as performance. Reported honestly — by breaking the ties on a declared random draw, which is what the lab does — it becomes the quantity everyone actually wanted: the capture a program would get by selecting customers at random, which is simply the selection share itself. That is the floor every candidate must clear, and it is a fact about random ordering rather than about the mean. The majority-class

baseline is the floor's categorical sibling, recorded here at concept level because this chapter's target is numeric; Chapter 9 will show it doing serious work, where “predict that nobody churns” turns out to embarrass many a classifier. The last-value baseline is the one with teeth in this chapter, because it is not a strawman — it is the incumbent. “Predict that the next six months look like the last six” is exactly the CRM operations lead's sorted spreadsheet, expressed as a prediction rule, and it is often strong: trailing spend genuinely carries signal, which is why the rule survives in so many businesses. The lab will discover its designed weakness — the miniature of Lab 8.1 shows it losing to the mean baseline on lumpy customers — but the weakness must be demonstrated, not asserted, and on the same field the model plays on.

Table 8.3

The three standard baselines, in StyleCraft form

Baseline	Rule	StyleCraft instance	What beating it proves
Mean	Predict the training mean for everyone	Every customer predicted the average spend_next6m; with ties broken on a declared random draw, its capture is the selection share — the random-ordering floor	The model learned something from customer differences
Majority class	Predict the most common class for everyone	(Categorical targets; operational in Chapter 9)	The model beats blind agreement with the crowd
Last value	Predict the previous window repeats	Each customer predicted their own July–December 2025 spend	The model beats the incumbent rule — and the sorted spreadsheet costs nothing

CONCEPT

The Baseline Is the Opponent

Every predictive deliverable in this guide reports at least two baselines beside the model, specified before the model exists — because a baseline chosen afterward can be chosen to lose. The practice has three payoffs. It converts model quality from an absolute into a margin, which is the only form a budget argument can use: “the model's list captures X dollars more than the sorted spreadsheet's list” prices the model; a naked MAE does not. It disciplines humility: on many marketing problems the last-value baseline is genuinely hard to beat, and knowing that early redirects effort from modeling to framing, where the leverage usually is.

And it brackets honest performance from both sides, which is the second half of the chapter's red-flag logic. A model performing far worse than every honest baseline has a defect or a broken frame — a mis-specified feature, a mismatched population, a target built at the wrong grain. A model performing implausibly better than every honest baseline warrants an urgent leakage audit. Remember which direction is which: for MAE and RMSE, lower is better, so “far below the baselines” is the suspicious result and “far above them” is the broken one. That sentence is worth reading twice, because the error metrics run the opposite way from the capture metric, where higher is better and the same two suspicions simply swap ends.

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013).

The section closes with the thread's forward arc stated once, in this chapter's terms: baselines began as interpretation (Section 2.7), operationalize here as opponents, extend in Chapter 10 to time — where naive and seasonal-naive rules inherit the last-value role — and take visual form in Chapter 12 as reference lines. One idea, four chapters, each adding machinery and none re-teaching it.

8.8 Overfitting, Underfitting, and the Bias–Variance Intuition

The holdout discipline of Section 8.5 exists to measure a gap; this section explains where the gap comes from and what its size means. Fit a sequence of models of increasing flexibility to the same training data — a straight line, then a curve, then a wigglier curve, then one flexible enough to pass near every point — and watch two error curves. Training error falls, or at least does not rise, with every step: more flexibility can only fit the fitting data better. Held-out error typically falls, then turns, then climbs. The turn is the most important shape in predictive modeling, and its two sides have names. A model on the left side, too simple to capture real structure — the mean baseline is the extreme case — is underfitting: it misses systematically, in the same direction for whole regions of customers, and both its training and its held-out errors are high. A model on the right side has begun learning its training data's noise: the accidental quirks, the specific whales, the coincidences that will not recur. It is overfitting, and its signature is the diverging pair — training error excellent and still improving, held-out error worsening — because every quirk memorized is a point of training flattery and a liability on customers whose quirks are different (James et al., 2021).

One precision about that first curve, because the lab will print it and the claim is easy to overstate. What is guaranteed, when nested polynomial terms are added to a least squares model, is that the training sum of squared errors cannot increase — least squares minimizes squared error, so a richer model can always reproduce the poorer one's fit and usually does better. Training RMSE therefore cannot rise with degree. Training MAE is a different summary of the same residuals, and nothing in the fitting criterion protects it: a richer model can reduce squared error while nudging the typical absolute miss slightly upward. The lab tracks training RMSE for the monotone claim and reports training MAE beside it as the quantity that generally improves without being required to.

DEFINITION

Overfitting and Underfitting

A model overfits when it captures patterns specific to its training data — noise, outliers, coincidence — that do not generalize, so that its training performance substantially exceeds its held-out performance and additional flexibility widens the gap. A model underfits when it is too simple to capture structure that does generalize, so that both training and held-out performance are poor. The two are diagnosed jointly from the training-versus-held-out error signature, not from either number alone.

Source: Adapted from James et al. (2021) and Domingos (2012).

The intuition underneath the U-shape is the bias–variance tradeoff, which this guide states at intuition level and uses forever after. A too-simple model is wrong on average — its errors have bias, systematic and shared across any sample you might have drawn. A too-flexible model is wrong differently every time — refit it on a different sample of customers and it memorizes different noise, so its predictions have variance, swinging with the accidents of the draw. Total error carries both, flexibility trades one for the other, and the honest minimum lies between the extremes — which is why “more sophisticated” is not a direction of improvement in predictive modeling, and why Domingos (2012) lists the belief that it is among the field's canonical beginner errors. Two of this guide's running facts sharpen the intuition. Sample size moves the turn: with more training customers, noise averages out more, and flexibility becomes safer — eight thousand customers tolerate a richer model than eight hundred. And irreducible uncertainty sets a limit: StyleCraft's spend labels contain genuine randomness — the wedding that did or did not happen in the outcome window — that no feature measured at the snapshot can carry.

That limit deserves a careful sentence, because it is easy to turn into a number the analysis has not earned. This guide does not estimate an irreducible error and does not claim to know where the limit sits. What it claims is weaker and more usable: individual human purchasing is volatile, spend is concentrated in a small number of customers, repeated windows of the same customer's history disagree with each other substantially, and honest baselines on this data land in a known range — so a result far stronger than all of that context would suggest is implausible on its face and belongs in the audit queue rather than in the deck. Implausible relative to domain

variability, outcome concentration, repeated-history stability, and the honest baselines: that is the standard, and it is a judgment supported by evidence rather than a threshold computed from a formula.

Table 8.4 collects the diagnostic signatures, because the pair of error numbers is the instrument the lab will actually read. The table's repair menu sits at the level this chapter needs: against overfitting — simplify the model, reduce the feature list to the frame's justified variables, or get more data; against underfitting — add real structure, which in this guide's practice usually means a feature the frame should have included, not a fancier algorithm. The menu's modesty is deliberate. The regularization machinery that automates the flexibility dial belongs to later coursework; what transfers to every future tool is the signature-reading skill. Note the column headings: during development the second number is a cross-validated estimate computed inside the training data (Section 8.9), not a test-set figure, because reading this table off the test set is how the test set gets spent.

Table 8.4

Reading the training-versus-validation error signature

Signature	Diagnosis	Response
Training error high, validation error high and similar	Underfitting — missing real structure	Add justified features or model structure; revisit the frame
Training error low, validation error much higher	Overfitting — memorizing training noise	Simplify; prune features; more data if available
Training and validation error both low and close together	Honest fit — the target zone	Report against baselines; proceed to the sealed test set, then Section 8.10
Validation error implausibly low relative to domain variability, outcome concentration, and the honest baselines	Too good — suspect leakage, not brilliance	Run the Section 8.11 audit before believing anything

8.9 Cross-Validation and the Discipline of Model Comparison

Section 8.5's protocol left a gap this section fills: development needs out-of-sample estimates too. Choosing between candidate models — the explanatory specification or a prediction-oriented one, the shorter or the longer feature list, a straight line or a curve — requires comparing their out-of-sample errors, but the test set is sealed until the end, and consulting it per candidate would spend it. The standard answer is cross-validation: rotate the holdout inside the training data. Partition the training customers into k equal folds — five is this guide's default; split at the unit of prediction, as always. Then, k times over, fit the candidate on all folds but one and evaluate it on the one left out, so every training customer serves once as validation and never validates a model that saw them. The k scores average into a single cross-validated estimate per candidate, with a bonus the single split cannot give: the spread across folds, which reads like every spread since Chapter 5 — a candidate whose error swings widely from fold to fold is telling you its performance depends on which whales it drew, the variance half of Section 8.8 made visible.

DEFINITION

Cross-Validation

Cross-validation estimates a model's out-of-sample performance using training data alone, by repeatedly refitting the model with one of k folds held out and averaging the k held-out scores. It is the standard instrument for development-time decisions — feature choices, model family, settings, and the amount of flexibility — allowing the true test set to stay sealed for the final grade. The averaged estimate remains honest only if every learned preprocessing step is refit inside each fold, and dishonest the moment any fold's validation data influences its own fitting.

Source: Adapted from James et al. (2021) and scikit-learn developers (2026b).

The definition's last two clauses are the reason this chapter's labs put every model inside a pipeline rather than fitting steps by hand. A pipeline is an object that chains learned steps — a standardizer, an imputer, a clustering model, the regression itself — and refits all of them together whenever it is fitted. Handed to a cross-validation routine, it is refitted inside each fold automatically, so the standardizer of fold three never sees fold three's validation customers and the segmentation of fold three never learned its centroids from them. Doing the same work by hand is possible and reliably forgotten under deadline; scikit-learn's own guidance is to use pipelines precisely so that this discipline is structural rather than remembered (scikit-learn developers, 2026a). The consequence for Chapter 6's segmentation is direct and worth stating before the lab reaches it: a segment label is a learned feature, not a raw column, so a segmentation refitted on all labeled customers before the split lets the test customers shape the scaler and the centroids that training then uses. Inside a pipeline, the same segmentation is honest.

What must be owned by the analyst is the comparison discipline cross-validation serves, because model comparison is where predictive projects most often go quietly wrong — not through any single dishonest act but through drift in the conditions of comparison. The discipline has three clauses, each closing a real loophole. Same split: every candidate is estimated on the same folds, generated once with a fixed seed, because reshuffling between candidates lets chance pick the winner — with a mediocre model and enough reshuffles, some split flatters it. Same metric: the metrics declared in the frame (Section 8.2) grade every candidate, with the decision metric leading; a candidate that wins on RMSE after losing on the declared capture and MAE has not won — the goalposts have moved, and moving goalposts after results are visible is the model-comparison version of writing the hypothesis after the data. Same baseline: the baselines of Section 8.7 appear in every comparison table, all the way to the final one, because the question is never only "which model is best" but "does the best model beat the spreadsheet by enough to matter."

The three clauses have a physical form in this chapter's labs: one leaderboard, built once, with a row per candidate and a column per declared metric, the two baselines occupying the first two rows, and every figure computed on the same five folds of the same training customers. Because the folds are shared, the leaderboard supports a comparison a table of separate averages cannot: the paired per-fold difference between a candidate and the incumbent. Pairing matters because folds differ in difficulty — a fold holding three whales is hard for everyone — so each candidate's own standard deviation across folds is inflated by difficulty the comparison should cancel. The honest question is not whether a candidate's mean beats the incumbent's mean by more than its own spread; it is whether the candidate beat the incumbent fold by fold. Only when the leaderboard is complete, and a selection rule written before the race has been applied to it, does the analyst name a finalist, refit it on all the training customers, and open the sealed test set — once. Everything before that moment is development; everything after it is reporting.

CONCEPT

One Leaderboard, Fixed Before the Race

Before any candidate is fitted, the analyst writes the leaderboard's rules — the folds, the metrics, the baselines, and the population — and every subsequent result joins that leaderboard unedited. The practice is predict-then-verify (Section 1.7) applied to model selection, and its enemy is not fraud but hope: the developer who has spent a week on a model wants it to win, and unfixed rules bend toward the wanted answer one small reasonable-seeming adjustment at a time.

Fixed rules also make delegation safe. An assistant asked to try ten model variants can be graded mechanically against the frozen leaderboard, which is exactly how Section 8.11 harnesses AI speed without inheriting AI judgment. When you read a claimed model comparison — a vendor's, a colleague's, an assistant's — the first question is not "what won?" but "when were the rules written, and was the test set part of the race or the finish line?"

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013) and scikit-learn developers (2026b).

8.10 From Score to Decision: Expected Value and Deployment Realities

A graded model is still not a decision. The model's output is a score — a predicted spend_next6m per customer — and the Backstage question is an action: which eight hundred customers get the treatment. This section builds the translation, and then follows the model past the decision into the part of its life that analytics courses traditionally ignore and analytics careers cannot: deployment.

The translation's first form is the one the fixed budget dictates: rank and cut. Score every eligible customer, sort descending, take the top eight hundred. Under a hard capacity constraint the model's job is ordering quality, not clairvoyance — it does not matter whether the top customer's prediction is 700 or 900 dollars if the ranking puts the right customers above the line — and the deliverable therefore leads with the capture exhibit Section 8.6 declared: what share of actual outcome-window spend the model's funded customers capture, against the share the incumbent's funded customers capture at the same capacity, with the count of customers appearing on both lists. That head-to-head, on sealed test customers, is the single most persuasive table the project produces, because it grades the exact artifact the meeting is deciding about. One operational detail belongs in the deliverable rather than in the code comments: at the cutoff, ties must break the same way every run, or the eight hundredth name changes between the analysis and the campaign build. The lab declares a secondary sort key for exactly this reason.

The translation's second form asks the question the budget's size begs: is eight hundred even the right number? Answering it requires attaching money to predictions, and the instrument is expected-value framing: for any candidate action on any customer, expected value is each possible outcome's value weighted by its probability, minus the action's cost, and the action is worth taking where the expectation is positive (Provost & Fawcett, 2013). At concept level — which is where this guide holds it, one notch above arithmetic and well short of decision theory — the framing does two jobs for the Backstage program. It converts the score threshold from an arbitrary rank into an economic line: treating a customer is justified where the treatment's expected incremental value exceeds twenty dollars. And it prices the two error types the opening case questions promised would differ: a wasted slot costs the treatment's twenty dollars; a missed high-spender costs the uncaptured margin of a customer the program existed to cultivate — an asymmetry in exactly Section 2.7's sense.

The word incremental in that paragraph is doing load-bearing work, and Section 8.1 flagged it in the brief so that it could be enforced here. This chapter's model predicts what a customer will spend. It does not predict what a customer will spend *because of the gift*, and no amount of out-of-sample accuracy converts the first quantity into the second. A perfectly calibrated spend model can rank the base beautifully and still send lookbooks to eight hundred customers who would have spent identically without one. That is not a flaw in the model; it is the boundary of what a predictive frame can establish, the same boundary Section 7.13's verb discipline enforced for coefficients, and only Chapter 11's instrument can cross it. The honest deliverable therefore prices the program two ways: the capture gap, which the analysis certifies, and the incremental return, which it explicitly does not — with a holdout design named as the purchase that would.

> **BRIDGE**
>
> Chapter 9 will make this asymmetry the whole instrument: when targets become yes/no, choosing the score threshold from the two error costs is threshold economics, and it is where the analyst meets the CFO.

Past the decision lies the deployed model's actual life, and this guide states its realities at concept level now so no student meets them first in production. A deployed model is a claim that the world it will score resembles the world it was trained on, and the world reliably breaks the claim in two distinct ways that deserve two distinct names. Data drift is change in the distribution or the quality of the inputs: StyleCraft's acquisition mix tilts suburban, a new store opens, the base gets younger, so the model is scoring customers whose feature combinations were rare in training. Concept drift is change in the relationship between inputs and outcome: the same recency, the same discount share, the same segment profile now imply a different level of future spending, because a pricing-and-packaging change restructured how customers buy (Gama et al., 2014). One term inside the first deserves precision, because it is often used as a synonym and is not one: covariate shift is the special case in which the predictor

distribution changes while the conditional relationship to the outcome holds steady. Data drift is the umbrella; covariate shift is its benign corner. The distinction earns its keep in monitoring: data drift can be detected by watching the inputs alone, and under covariate shift a model may keep performing perfectly well, while concept drift cannot be diagnosed from input distributions at all and can degrade performance without moving a single input.

Performance decay is the observable consequence of either — measured predictive quality eroding as one or both accumulate, sometimes gradually, sometimes at a policy change's step edge. The disciplines that answer them follow directly. Monitoring means tracking the deployed model's realized error on each period's fresh labels as they mature, against the benchmark the test set established, exactly as the test set once was — and, cheaply and separately, watching the input distributions, because an input alarm arrives months before an outcome alarm can. Retraining runs on a declared cadence tied to the horizon: a six-month-horizon model can only be fully graded six months after each scoring run, so the monitoring calendar is part of the frame, written before deployment, not improvised after the first bad quarter. And explicit triggers — a performance floor, a known structural change such as a repricing or a merchandising overhaul — force revalidation off-cadence, because concept drift's most expensive property is that it announces itself in the business before it announces itself in the metrics. Sculley et al. (2015), in the paper that named the field's hidden costs, put the engineering point this guide converts to an analyst's point: the model is the smallest part of the system, and the system — data feeds, definitions, monitoring, retraining, ownership — is what actually predicts. The analyst of record's obligations do not end at the test set. They end when the model is retired.

DEFINITION

Data Drift, Concept Drift, and Performance Decay

Data drift is post-deployment change in the distribution or quality of a model's input features, so that scored cases increasingly differ from training cases; covariate shift is its special case, in which the predictor distribution changes while the conditional relationship to the outcome remains stable. Concept drift is change over time in the relationship between the inputs and the target, so that the learned mapping no longer describes how the outcome is generated, and it cannot be diagnosed from the inputs alone. Performance decay is the resulting erosion of predictive quality, observed by monitoring realized error as fresh outcomes mature. All three are managed rather than prevented: by monitoring inputs and outcomes against the training-window benchmark, retraining on a cadence tied to the horizon, and declaring triggers that force revalidation when a structural change is known.

Source: Adapted from Gama et al. (2014) and Sculley et al. (2015).

8.11 AI as a Predictive-Modeling Assistant

Chapter 7's AI section ended with a division of labor: the assistant proposes relationships; the analyst tests whether they are honestly stated. Predictive work sharpens the division again, because the assistant's fluency now extends end to end. Prompted with a feature table and “build me a model that predicts customer spend,” a current assistant will construct the label, engineer features, split the data, fit several model families, tune them, and report a polished comparison table — an entire pipeline, delivered in one response, runnable as pasted. The productivity is real and this guide uses it. The hazard is equally real and has a precise shape: every failure mode this chapter has named is a failure the assistant commits fluently, silently, and with congratulations. A leaky pipeline does not look different from an honest one in the chat window. It looks better.

The characteristic failure modes, named so the audit can target them. The convenient-table leak: asked to predict future spend, the assistant builds features from whatever table it was given — including columns computed across the outcome window — because nothing in the prompt drew the snapshot wall; temporal leakage by default, Table 8.2's first row, and the single most common defect in AI-drafted pipelines. The celebration of the impossible: the leaky pipeline reports R^2 of 0.97, and the assistant's narration calls it excellent — assistants inherit their training

data's enthusiasm for high fit statistics, not this chapter's suspicion of them, so the too-good reflex is precisely the judgment that does not transfer. The missing opponent: the report quotes MAE, RMSE, and R^2 with no baseline anywhere, leaving no way to know whether the model beats the sorted spreadsheet it exists to replace. The metric mismatch: distance metrics reported for a ranked-list decision, with no capture figure and no head-to-head against the incumbent's list, because nothing in the prompt named the act. The vocabulary shuffle: “96 percent accuracy” applied to a numeric target — accuracy is classification vocabulary (Chapter 9's, where its own failures await), and its appearance in a numeric-prediction claim is a tell that the claimant is quoting whatever number sounded best. The hygiene slips: preprocessing fitted on pooled data, the test set consulted in a tuning loop, complexity chosen by watching test error, the comparison metric switching mid-report — each individually small, each a clause of Sections 8.5 and 8.9 violated without announcement. And the deletion helpfulness: zero-spend labels “cleaned” away as missing data, quietly deleting the customers the model most needs to learn (Section 8.3).

Where the assistant genuinely helps, use it deliberately. Pipeline mechanics from a written frame — the split with a stated seed, the fold generator, the metric functions, the baseline estimators, the leaderboard table — are exactly the well-specified boilerplate assistants draft reliably. Feature derivation code is safe when, and only when, the prompt supplies the derivation rules and the window boundaries explicitly; the lab's prompts do. Audit exhibits — the feature-window trace, the range checks of Section 8.4, the signature table of Section 8.8 — are tedious, mechanical, and ideal for delegation. Error analysis is a strength: asked “which customers does this model miss worst, and what do they share?”, an assistant surveys the residuals faster than any manual pass, and the answer — on StyleCraft's data, the lumpy occasion shoppers — feeds the next framing conversation. And the assistant remains a good rehearsal partner for the vendor meeting.

The audit that governs all of it extends Chapter 7's four points to the predictive setting. Five points, in order, run on every model — the assistant's, the vendor's, and yours — before any result is believed or repeated. Table 8.5 states the checklist; the box operationalizes it as the two-prompt pattern this guide has used since Chapter 6, with the audit standing between mechanics and narration.

Table 8.5

The five-point audit of a predictive model

Point	The check	Fails when
1. Frame	Unit, target, features, horizon, metrics, and baselines declared in writing; label defined with snapshot, both window boundaries, filters, and eligibility	The frame is implicit, or reconstructed after the results
2. Leakage	Every feature traced to its window and derivation; range checks for impossible values; Table 8.2's five routes checked	Any feature crosses the snapshot, derives from the label, or cannot be traced
3. Split hygiene	Split at the unit of prediction, seeded; every learned step inside a pipeline fitted on training folds only; complexity chosen by cross-validation; test set consulted once	The test set steered development, or a learned step saw pooled data
4. Baseline	Mean and last-value baselines on the same folds and the same test set, on every declared metric; the model's margin stated	No baseline reported, or the baseline chosen after the results
5. Error in decision units	The declared decision metric leads; MAE and RMSE read in dollars beside it; the result checked for plausibility against domain variability, outcome concentration, and the baselines	Metrics shuffle mid-report, or excellence is celebrated instead of audited

AI IN PRACTICE

The Pipeline That Must Survive Its Audit

A two-prompt pattern with the audit between, extending Section 7.11's. Prompt one, the frame as specification: paste the seven declarations of Table 8.1, the label definition with its four dates and eligibility rule, the feature list with derivation rules and the instruction that every feature be computed from transactions inside the declared feature window, the declared metrics and baselines, and the split and cross-validation protocol with its seed — then ask for mechanics only: “Build this pipeline exactly as specified; select among the candidates using five-fold cross-validation on the training customers only; report one leaderboard with capture, MAE, and RMSE for every candidate and both baselines; do not touch the test set; change nothing in the specification; narrate nothing yet.”

Then run Table 8.5 yourself, in order: confirm the frame came back unedited; trace three features to their windows and run the recency range check; confirm every learned step sits inside a pipeline and the test set is untouched; confirm both baselines are present on every metric and the margin is computed; read the signature against Table 8.4, and if anything looks too good, stop and trace. Prompt two, only after the pipeline survives and the finalist has been graded once on the test set: “Draft the comparison paragraph for a marketing VP: the model's capture at the program's capacity against the incumbent list, its test MAE against both baselines in dollars, and no claim that any predicted difference is caused by anything.” Audit the draft's verbs against Section 7.2's registry — prediction language earns “expected” and “predicted,” never “will drive” — and file both exchanges per Appendix D. Predictions first, per Section 1.7; the analyst of record signs the scores.

Source: Course concept developed for this guide, informed by scikit-learn developers (2026a) and Kaufman et al. (2012).

8.12 Hands-On Application in Python and Google Colab

The preceding sections built the disciplines; this section runs them against the Backstage decision, in two labs that mirror the chapter's argument. Lab 8.1 builds the frame's raw material and the opponents: the miniature by hand, then the snapshot-safe feature table, the two windows, the eligible labeled population, and the split that seals the test customers. Lab 8.2 runs the race: a leaderboard of candidates cross-validated inside the training data, an overfitting demonstration conducted without ever consulting the test set, the finalist opened against the sealed customers once, a deliberate leak built and caught, and the score translated into the list. The labs use the certified transactions, customers, and products files; the feature table is rebuilt from Table 6.5's derivation rules with the analysis date moved back to the snapshot. AI assistants may draft any code cell (Appendix C has templates; Appendix A covers Colab mechanics); every output is predicted before it is computed, and every exchange is documented per Appendix D.

Two conventions hold across every cell, and both are this chapter's disciplines made structural. Every model, including every baseline, is an estimator evaluated through the same cross-validation call on the same folds, so that “same split, same metric, same baseline” is enforced by the code rather than remembered by the analyst — the labs use scikit-learn (Pedregosa et al., 2011) throughout. And every step that learns anything from data — a standardizer, a segmentation, the regression itself — lives inside a pipeline, so that it is refitted within each training fold and never estimates a single quantity from a customer it is about to be graded on (scikit-learn developers, 2026a).

8.12.1 Lab 8.1, Part A: Baselines by Hand in Miniature

The miniature is the ten-customer table Labs 6.1 and 7.1 built intuition on, now given the chapter's time structure: for each customer, spend in the six months before the snapshot (spend_prior6m, the feature side) and spend in the six months after (spend_next6m, the label). The values are rounded to whole dollars for hand work and consistent

with the miniature's running profiles: the steady Urban Loyal Core customers repeat, the occasion shoppers lump, the lapsed customer stays dark, and one customer arrives almost from nowhere. Before running anything, predict which baseline wins on these ten customers — the mean, or the incumbent's last-value rule — and write one sentence of reasoning.

Code 8.1. Calculate the miniature baselines

```
import pandas as pd

mini = pd.DataFrame({
    "customer_id": ["C001","C002","C003","C004","C005",
                    "C006","C007","C008","C009","C010"],
    "spend_prior6m": [90.0, 400.0, 30.0, 0.0, 60.0,
                      0.0, 24.0, 70.0, 0.0, 0.0],
    "spend_next6m":  [80.0, 0.0, 30.0, 44.0, 60.0,
                      0.0, 0.0, 50.0, 38.0, 312.0],
}).set_index("customer_id")

mean_pred = mini["spend_next6m"].mean()   # mean baseline
lv_pred   = mini["spend_prior6m"]         # last-value baseline

err_mean = mini["spend_next6m"] - mean_pred
err_lv   = mini["spend_next6m"] - lv_pred

mae_mean, mae_lv   = err_mean.abs().mean(), err_lv.abs().mean()
rmse_mean = (err_mean ** 2).mean() ** 0.5
rmse_lv   = (err_lv ** 2).mean() ** 0.5

print(f"mean:       MAE {mae_mean:6.2f}  RMSE {rmse_mean:6.2f}")
print(f"last-value: MAE {mae_lv:6.2f}  RMSE {rmse_lv:6.2f}")

# The decision metric on ten rows: with one slot, who does each rule pick,
# and what share of the ten customers' next-six-month spend does it capture?
total = mini["spend_next6m"].sum()
ranked  = mini.sort_values("spend_prior6m", ascending=False)
pick_lv = ranked["spend_next6m"].iloc[0]
print(f"one-slot capture, last-value rule: {pick_lv / total:.1%}")
best = mini["spend_next6m"].max()
print(f"one-slot capture, perfect ranking: {best / total:.1%}")
```

Expected output: two lines of MAE and RMSE, then two capture figures showing what a single-slot program would have captured under the incumbent rule and under a perfect ranking.

Input: ten literal customers. Transformation: two prediction rules, three metrics. Output: the leaderboard of Section 8.7 at a scale where every figure can be checked by hand. The last two lines are the point of the cell's revision from a purely calibration-based exercise: they ask the ranking question rather than the distance question, and on these ten customers the two questions have spectacularly different answers.

VERIFICATION CHECK

Before you run: compute both baselines by hand. The label mean is 614 ÷ 10 = 61.40 dollars. The mean baseline's ten absolute errors (18.60, 61.40, 31.40, 17.40, 1.40, 61.40, 61.40, 11.40, 23.40, 250.60) sum to 538.40, so its MAE is 53.84; its squared errors average 7,642.44, so its RMSE is 87.42. The last-value baseline's errors (10, 400, 0, 44, 0, 0, 24, 20, 38, 312) sum to 848, so its MAE is 84.80; its squared errors average 26,180, so its RMSE is 161.80. Then predict the capture lines: if the program had one slot and filled it by trailing spend, which customer would receive it, and how much of the ten customers' next-window spend would that customer account for?

After you run: reconcile every figure to the cent, then confront two results most students do not predict. The first is that the incumbent rule loses to the crudest baseline in the book on both distance metrics — and loses while being the better predictor for half the customers. For the steady core (C001, C003, C005) and the dark customer who stays dark (C006), yesterday repeats and last-value is nearly perfect. The rule is destroyed by four customers: C002, whose 400-dollar November baskets do not recur; C010, whose 312-dollar occasion arrives from a prior spend of zero; and C004 and C009, who ignite from nothing. This is Chapter 5's lumpiness finding wearing prediction clothes, and the RMSE gap — 161.80 against 87.42, proportionally wider than the MAE gap — is Section 8.6's concentration alarm ringing on ten rows. The second result is sharper: the incumbent's one slot goes to C002, who spends nothing at all in the outcome window, so the rule captures zero percent of what a single perfect pick would have captured. Distance and decision are different questions, and the second one is the program's. Write the implication in one sentence: a model earns its budget line only if it can keep last-value's wins on the steady customers while fixing its catastrophes on the lumpy ones — and the features that could do that (occasionwear share, purchase rhythm, engagement) are precisely what the sorted spreadsheet ignores.

Investigate if: your hand arithmetic disagrees with the printed figures anywhere. Every number in this cell is reproducible with a calculator, and a mismatch is arithmetic rather than convention — unlike Chapter 7's covariance denominators, nothing here depends on a software default.

8.12.2 Lab 8.1, Part B: The Windows, the Labels, and the Eligible Population

Part B builds the real thing. Begin by writing the frame — the seven declarations of Table 8.1, the label definition with its four dates, the eligibility rule, and the split protocol — into a markdown cell at the top of the notebook. The frame precedes the code, and the lab is graded on that order.

Then build the feature side of the wall. Rather than describing the rebuild in a comment, the lab supplies it as a function, because a chapter distributed on its own cannot assume the reader has Lab 6.1's notebook open, and because a feature builder that takes its window as an argument is the only kind that can be pointed at a snapshot without being edited. Code 8.2 is Table 6.5's derivation rules with the analysis date and window start promoted to parameters. The companion repository ships the identical function as stylecraft_features.build_customer_features, so a student who prefers the import can use it and get the same columns; the version printed here exists so that nothing in the pipeline is a black box.

Code 8.2. Roll up the feature window to customer grain

```
import numpy as np
import pandas as pd

# The four dates the frame declared. Every window has two ends, and both
# ends are enforced in code rather than trusted to the file's contents.
FEATURE_START = pd.Timestamp("2024-07-01")
SNAPSHOT      = pd.Timestamp("2025-12-31")
OUTCOME_START = pd.Timestamp("2026-01-01")
OUTCOME_END   = pd.Timestamp("2026-06-30")


def snapshot_rollup(transactions, customers, products,
                    snapshot, window_start):
    """Lab 6.1's aggregation, parameterized by window rather than
    hard-coded. One row per customer with at least one order inside
    [window_start, snapshot]. Nothing dated after the snapshot can reach
    the output, because nothing after it is ever read. Normalizing here
    rather than in the caller keeps a late-in-the-day December 31 order
```

```
    inside the feature window."""
    tx = transactions.assign(
        order_date=transactions["order_date"].dt.normalize())
    tx = tx[tx["order_date"].between(window_start, snapshot,
                                     inclusive="both")]

    # "min" and "first" are honest only if the attribute is constant
    # within the order. Check, do not assume (Chapter 4's rule).
    for col in ["customer_id", "order_date", "channel"]:
        assert (tx.groupby("order_id")[col].nunique(dropna=False)
                  .eq(1).all()), f"an order carries multiple {col} values"

    orders = (tx.groupby(["order_id", "customer_id"], as_index=False)
                .agg(order_date=("order_date", "min"),
                     order_revenue=("line_revenue", "sum"),
                     channel=("channel", "first")))
    # Shares live at line grain (Section 6.6), not in order totals.
    lines = tx.merge(products[["product_id", "is_occasionwear"]],
                     on="product_id", how="left", validate="many_to_one")
    assert lines["is_occasionwear"].notna().all(), "unmatched product_id"
    shares = (lines
              .assign(
                  disc=lambda d: d["line_revenue"]
                                   .where(d["discount_pct"] > 0, 0.0),
                  occ=lambda d: d["line_revenue"]
                                  .where(d["is_occasionwear"], 0.0))
              .groupby("customer_id")
              .agg(discounted_revenue=("disc", "sum"),
                   occasion_revenue=("occ", "sum")))
    return (orders.groupby("customer_id")
                  .agg(last_order=("order_date", "max"),
                       frequency=("order_id", "nunique"),
                       monetary=("order_revenue", "sum"),
                       store_orders=("channel",
                                     lambda s: (s == "Store").sum()))
                  .join(shares)
                  .join(customers.set_index("customer_id")
                                 [["signup_date"]]))
```

Expected output: nothing — this cell declares the four dates and defines a function.

Input: the certified line-grain, customer, and product files. Transformation: a window filter, an order-grain roll-up, and two line-grain shares. Output: one row per customer with at least one order inside the declared window, carrying the raw counts and sums that Code 8.3 turns into features. The split between this cell and the next mirrors Chapter 6's own Code 6.5 and Code 6.6, for the same reason: aggregation and derivation fail differently and are checked differently.

The function's most important property is negative: it never reads a transaction outside the window it was given, so temporal leakage cannot enter through it no matter what the caller does downstream. That is why the window filter is the first statement rather than a later convenience — and why the same function will be used in Code 8.13 to manufacture the leak, by handing it a later snapshot, which is exactly how the error happens in practice.

Code 8.3. Derive the ratio and intensity features at the snapshot

```
def build_customer_features(transactions, customers, products,
                            snapshot, window_start):
    """Table 6.5's ratio and intensity derivations, computed at a declared
    snapshot. Indexed by customer_id; every column is snapshot-safe by
    construction, because snapshot_rollup read nothing later."""
    cf = snapshot_rollup(transactions, customers, products,
                         snapshot, window_start)
```

```
    signup = cf["signup_date"].clip(lower=window_start)
    observed_days = (snapshot - signup).dt.days
    cf["exposure_months"]    = np.maximum(observed_days / 30.44, 1.0)
    cf["recency_days"]       = (snapshot - cf["last_order"]).dt.days
    cf["tenure_days"]        = (snapshot - cf["signup_date"]).dt.days
    cf["aov"]                = cf["monetary"] / cf["frequency"]
    cf["orders_per_month"]   = cf["frequency"] / cf["exposure_months"]
    cf["revenue_per_month"]  = cf["monetary"] / cf["exposure_months"]
    cf["discount_share"]     = cf["discounted_revenue"] / cf["monetary"]
    cf["store_share"]        = cf["store_orders"] / cf["frequency"]
    cf["occasionwear_share"] = cf["occasion_revenue"] / cf["monetary"]

    flags = (customers.set_index("customer_id")
                      [["app_user", "email_opt_in"]].astype(int))
    return (cf.join(flags)
              .drop(columns=["last_order", "signup_date", "store_orders",
                             "discounted_revenue", "occasion_revenue"]))
```

Expected output: nothing — this cell defines the function the next one calls.

Input: the roll-up. Transformation: exposure, recency, tenure, the three shape ratios, and the two intensity rates, all measured against the snapshot rather than against a hard-coded analysis date. Output: the customer feature table as StyleCraft could have built it on December 31, 2025, and nothing else.

Every derivation here is Table 6.5's, unchanged in substance and changed in one respect that matters: the date it counts back from is an argument. A feature builder that hard-codes its analysis date cannot be pointed at a snapshot without being edited, and a pipeline whose feature step must be edited by hand between runs is a pipeline that will eventually be run with the wrong date. Parameterizing the window is a leakage control, not a matter of style.

Code 8.4. Rebuild the feature table and check its contract

```
# The certified files are loaded per Appendix A as transactions_clean,
# customers, and products. Check the INPUTS before building anything:
# a builder cannot repair a dimension table with duplicate keys.
assert customers["customer_id"].is_unique
assert products["product_id"].is_unique
assert transactions_clean[["order_id", "customer_id",
                           "order_date"]].notna().all().all()

cf_snap = build_customer_features(transactions_clean, customers,
                                  products, snapshot=SNAPSHOT,
                                  window_start=FEATURE_START)

# A feature contract stated in code fails loudly when an upstream build
# changes; a contract stated in prose fails silently, six cells later.
EXPECTED = {"frequency", "monetary", "exposure_months", "recency_days",
            "tenure_days", "aov", "orders_per_month", "revenue_per_month",
            "discount_share", "store_share", "occasionwear_share",
            "app_user", "email_opt_in"}
assert set(cf_snap.columns) == EXPECTED, "the feature contract changed"
assert cf_snap.index.is_unique
assert cf_snap.notna().all().all(), "a feature is undefined"
assert (cf_snap["recency_days"] >= 0).all()
tol = 1e-9
for share in ["discount_share", "store_share", "occasionwear_share"]:
    assert cf_snap[share].between(-tol, 1 + tol).all(), share

print("customers with a feature-window purchase:", len(cf_snap))
print(cf_snap[["recency_days", "frequency", "monetary", "aov",
               "discount_share", "occasionwear_share"]].describe().round(2))
```

Expected output: a customer count for the feature window, then a six-column summary in which recency is never negative and every share lies between 0 and 1; eleven assertions pass silently.

Input: the certified files and the two functions above. Transformation: three input checks, one call, and eight output checks. Output: cf_snap, the snapshot-safe feature table, plus the evidence that it is what it claims to be. The checks run in both directions on purpose. Before the build, the cell confirms what the builder cannot repair: duplicate keys in a dimension table would silently multiply rows through the joins, and a missing order identifier would quietly drop an order from a customer's history. After the build, the column-set assertion is the guard worth copying into every project: a feature contract stated in code fails loudly when an upstream build changes, where a contract stated in prose fails silently six cells later, in a model whose numbers look fine.

Code 8.5. Enforce both windows and construct the labeled eligible table

```
# Dates are recorded at day grain; normalizing makes the boundary
# comparisons exact rather than dependent on a stored time component.
tx = transactions_clean.assign(
    order_date=transactions_clean["order_date"].dt.normalize())

in_feat = tx["order_date"].between(FEATURE_START, SNAPSHOT,
                                   inclusive="both")
in_out  = tx["order_date"].between(OUTCOME_START, OUTCOME_END,
                                   inclusive="both")
tx_feat, tx_out = tx[in_feat], tx[in_out]

# Both ends of both windows, asserted rather than assumed.
assert tx_feat["order_date"].min() >= FEATURE_START
assert tx_feat["order_date"].max() <= SNAPSHOT
assert tx_out["order_date"].min()  >= OUTCOME_START
assert tx_out["order_date"].max()  <= OUTCOME_END
assert len(tx_feat.index.intersection(tx_out.index)) == 0, \
    "the two windows overlap"

labels = (tx_out.groupby("customer_id")["line_revenue"]
                .sum().rename("spend_next6m"))

# The incumbent's own column: the trailing six months of the feature window.
prior6 = (tx_feat[tx_feat["order_date"] >= pd.Timestamp("2025-07-01")]
          .groupby("customer_id")["line_revenue"]
          .sum().rename("spend_prior6m"))

# ELIGIBILITY (Section 8.3), both conditions evaluable at the snapshot:
# signed up on or before it, and at least one purchase inside the window.
signed_up = set(customers.loc[customers["signup_date"] <= SNAPSHOT,
                              "customer_id"])
purchased = set(cf_snap.index)
eligible  = sorted(signed_up & purchased)

model_df = (cf_snap.loc[eligible]
            .join(labels)
            .join(prior6)
            .fillna({"spend_next6m": 0.0, "spend_prior6m": 0.0}))

print("customers in the file:              ", len(customers))
print("signed up by the snapshot:          ", len(signed_up))
print("purchased inside the feature window:", len(purchased))
print("eligible (both conditions):         ", len(model_df))
print("signed up, no feature-window purchase (activation population):",
      len(signed_up - purchased))
print("acquired after the snapshot -- outside the population this",
      "training snapshot can support:", len(customers) - len(signed_up))
```

Expected output: six counts that reconcile — customers in the file, signed up by the snapshot, purchasers in the feature window, eligible customers, the activation population, and the post-snapshot acquisitions — and five assertions passing silently.

Input: the certified transactions and customers, plus the snapshot feature table. Transformation: two window filters with both boundaries enforced, one label aggregation, one trailing-window aggregation, and a two-condition eligibility rule. Output: the modeling table, one row per eligible customer, with features on one side of the wall and the label on the other.

Three details carry the section's arguments. The window filters name four dates rather than one, so the feature window cannot silently absorb history older than the declaration and the outcome window cannot silently absorb months beyond the horizon; the assertions then check both ends of both, and the intersection check proves the two windows share no transaction. The eligibility rule intersects two conditions, both evaluable at the snapshot, which is what keeps it clear of Table 8.2's population-leakage row. And the printed counts partition the customer file exactly, in the manner Chapter 6 established for its clustering base: the eligible customers, the signed-up customers with no feature-window purchase — an activation population, not a modeling population — and the customers acquired after the snapshot, whom this model was not built to score. Reporting the excluded groups by name is the difference between a declared population and a convenient one.

The fillna line deserves its own sentence, because deleting it would be the most consequential one-character edit in the lab. A customer eligible at the snapshot who bought nothing between January and June has no row in the labels series, so the join produces a missing value — and that missing value is not missing data. It is a label of zero, the outcome the model most needs to be able to anticipate (Section 8.3).

Code 8.6. Audit the labeled table before splitting

```
assert model_df.index.is_unique
assert model_df.notna().all().all(), "a feature or label is missing"
assert (model_df["spend_next6m"] >= 0).all()
assert (model_df["spend_prior6m"] >= 0).all()
assert (model_df["recency_days"] >= 0).all()
assert (model_df["tenure_days"] >= model_df["recency_days"]).all(), \
    "a customer purchased before signing up -- check the join"

# An exact count, not a quantile: ties at the 99th percentile can sweep in
# far more than one customer in a hundred.
n_top = max(1, int(np.ceil(len(model_df) * 0.01)))
top1 = (model_df["spend_next6m"].nlargest(n_top).sum()
        / model_df["spend_next6m"].sum())

# The program's capacity is a budget, not a proportion: $20 a slot and a
# fixed program budget fund a fixed number of customers. The selection
# share is that capacity divided by the population actually scored, and
# it is the share every capture figure in this chapter uses.
CAPACITY = 800
SELECTION_SHARE = CAPACITY / len(model_df)

print(f"labeled customers:            {len(model_df):,}")
print(f"zero-label share:             "
      f"{model_df['spend_next6m'].eq(0).mean():.1%}")
print(f"top {n_top} customers (1% by count) hold "
      f"{top1:.1%} of outcome spend")
print(f"program capacity:             {CAPACITY} slots")
print(f"selection share:              {SELECTION_SHARE:.1%}")
print(model_df["spend_next6m"].describe().round(2))
```

Expected output: the labeled row count, the zero-label share, the share of outcome spend held by the top one percent of customers, and an eight-row summary of the label; six assertions pass silently.

Input: the modeling table. Transformation: none — this cell computes nothing the model uses. Output: the evidence that the table means what the frame says it means. The assertions encode claims the prose has been making: labels are non-negative because they are sums of revenue, no feature is undefined because the eligibility rule guaranteed a denominator, and no customer purchased before signing up, which would indicate a broken join rather than an unusual customer.

The two printed shares are calibration for the too-good reflex, and they are worth recording in the notebook before any model exists. A base in which a quarter of eligible customers spend nothing in the outcome window, and in which the top one percent of customers hold a large share of the spend that does occur, is a base whose individual outcomes are volatile by construction. That is the context Section 8.8 said should replace an invented noise floor: not a computed limit, but a documented picture of how much of this target is genuinely unpredictable from anything measured six months earlier.

Code 8.7. Split the customers and freeze the test set

```
from sklearn.model_selection import train_test_split

train_df, test_df = train_test_split(model_df, test_size=0.20,
                                     random_state=42)
y_tr, y_te = train_df["spend_next6m"], test_df["spend_next6m"]

# Identities, not positions: every later comparison reuses these.
train_ids, test_ids = train_df.index, test_df.index
assert train_ids.intersection(test_ids).empty
assert len(train_ids) + len(test_ids) == len(model_df)

# Development diagnostics come from the TRAINING customers only. The test
# labels are stored above and deliberately not summarized here: knowing
# that the test half holds fewer whales, or a different zero rate, is
# knowledge that would leak into feature choices and expectations.
print(f"training customers: {len(train_ids):,}")
print(f"test customers:     {len(test_ids):,}")
print(f"training zero-label share: {y_tr.eq(0).mean():.1%}")
print(f"training mean / median:    {y_tr.mean():,.2f} / "
      f"{y_tr.median():,.2f}")
print(f"training 99th percentile:  {y_tr.quantile(0.99):,.2f}")
print("\nTEST SET SEALED. Its outcomes stay unopened until Code 8.14.")
```

Expected output: two identity assertions, a six-row comparison of the training and test halves, and the sealing notice.

Input: the audited modeling table. Transformation: a seeded 80/20 split at the customer grain. Output: two frames and, more importantly, two index objects — train_ids and test_ids — which every later cell reuses. Identities rather than positions is the point: a random seed reproduces which positions go where, not which customers, so any later comparison that re-splits a differently ordered frame can quietly grade two models on two populations. Code 8.13 depends on this.

Notice what this cell does not print, because the omission is the discipline. It would be natural to compare the two halves here — zero-label shares, means, upper percentiles — to reassure yourself that the split was fair. That comparison reads the test labels, and a sealed test set is not sealed once its outcomes have been seen. Learning that the test half holds fewer whales, or a gentler mean, is knowledge that leaks forward into every later choice: which features feel necessary, how much flexibility feels safe, what result will feel plausible. None of that requires bad faith; it requires only a human who now knows something. The halves are compared in Code 8.14, after the grade is recorded, where the same numbers are context for reading a result rather than an influence on producing one.

What the cell prints instead is the training half's own profile, which is available without cost because the training labels are development data by definition. If the split were badly unbalanced, the cross-validated spreads of Code 8.11 would show it — folds drawn from a lumpy training half disagree with each other — and that signal, unlike a peek at the test set, is one the protocol permits.

8.12.3 Lab 8.2, Part A: The Candidates and the Cross-Validated Leaderboard

Lab 8.2 runs the race Section 8.9 specified. Five candidates enter, and the two baselines are candidates rather than commentary: they are fitted, scored, and ranked by the identical call on the identical folds, because a baseline evaluated by a different route is not evidence about the same question. The three models are chosen to make an argument the chapter has been building toward.

The first is the explanatory-style candidate: Chapter 7's interpretable predictors — recency, tenure, app use, email opt-in, discount share — refitted to the new frame, with behavioral cluster indicators standing where Chapter 7 put its segment dummies. Its presence is the chapter's continuity argument and its own quiet correction. Continuity, because the same equation family is being asked to do a different job under a different grade. Correction, because carrying that specification into a predictive frame exposes something the drivers analysis never had to think about: a segment label is not a raw column but a learned feature, produced by a scaler and a set of centroids estimated from data. Refit that segmentation on all labeled customers before the split and the test customers have shaped the very definitions the training data is described by — Table 8.2's split-contamination row, committed by a step nobody thinks of as a model. The lab's answer is structural: the clustering becomes a pipeline step, refitted inside every fold, assigning rather than learning from the customers it scores.

That repair costs something, and the chapter is explicit about the price rather than smuggling it. A clustering refitted inside each fold cannot carry Chapter 6's four named segments, because naming them was an act of judgment over full centroid profiles, performed once, on one estimation. What survives fold to fold is an ordering rule, not a nameplate — so the candidate's indicators say something like second-lowest purchase intensity rather than Suburban Occasion, and two clusters carrying the same rank in different folds can differ materially in discount behavior or basket shape. The candidate is therefore called explanatory-style rather than the Chapter 7 specification. Recovering the named segments honestly would take a frozen assignment rule, established before this study and applied unchanged to every fold; that is a reasonable engineering answer and a different lab. Naming the limitation is this one's job.

The second is the prediction-oriented candidate, and the reason it exists is the sharpest methodological point in the lab. Chapter 7 chose its variables to make coefficients defensible. Nothing about that criterion selects the strongest predictive inputs, and the specification it produced omits, for prediction, almost everything a snapshot legitimately knows: feature-window spend, purchase frequency, average order value, orders and revenue per month, store and occasionwear mix — and the incumbent rule's own trailing-spend column, which belongs in the model precisely because a good model may beat the spreadsheet by keeping its signal and combining it with everything the spreadsheet ignores. Every one of those columns is computed inside the feature window, so every one of them is snapshot-safe; excluding them would not be caution but a category error, applying an explanatory chapter's variable-selection rule to a predictive chapter's job.

The third is a deliberately more flexible version of the second — the same features expanded to quadratic terms — so the leaderboard contains at least one candidate whose extra flexibility might or might not pay for itself. Whether it does is the leaderboard's business, not the analyst's hope, and Part B examines the general shape of that question.

Code 8.8. Express the two baselines as estimators

```
import numpy as np
from sklearn.base import BaseEstimator, RegressorMixin, TransformerMixin
from sklearn.cluster import KMeans
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import PolynomialFeatures, StandardScaler


class MeanBaseline(BaseEstimator, RegressorMixin):
    """Predict the training mean for everyone.

    A constant score carries no ranking information at all, so its
```

```
    capture would be decided entirely by whatever order the rows arrive
    in. Rather than let row order masquerade as a result, this baseline
    adds a seeded, financially negligible jitter: the dollar metrics are
    unchanged to the cent, and the ranking becomes explicitly random --
    which is what the capture floor actually is."""

    def __init__(self, random_state=42):
        self.random_state = random_state

    def fit(self, X, y):
        self.mean_ = float(np.mean(y))
        return self

    def predict(self, X):
        rng = np.random.default_rng(self.random_state)
        return self.mean_ + rng.normal(0.0, 1e-9, len(X))


class LastValueBaseline(BaseEstimator, RegressorMixin):
    """The incumbent, expressed as an estimator: predict that the trailing
    six months repeat. It fits nothing, so it can never leak anything."""

    def fit(self, X, y=None):
        return self

    def predict(self, X):
        return np.asarray(X, dtype=float).ravel()
```

Expected output: nothing — this cell defines two classes.

Input: none. Transformation: two small estimator definitions. Output: objects a cross-validation routine can fit, score, and refit per fold exactly as it does a regression. Read them rather than rewriting them.

LastValueBaseline is the incumbent expressed as an estimator: its fit method does nothing, which is the honest characterization of a rule that learns nothing, and it is therefore structurally incapable of leaking. Expressing the baseline this way is not decoration — it is what allows the same cross-validation call, on the same folds, with the same metrics, to produce the baseline's row of the leaderboard, which is the same-split clause of Section 8.9 enforced by construction rather than by discipline.

MeanBaseline carries the correction Section 8.7 argued for, and its docstring is the lesson. Predicting one number for everyone is a perfectly good calibration baseline and an empty ranking: with every score identical, which customers land in the funded tenth is decided by the order the rows happen to be in, and a sorting routine will report that accident as a capture figure without complaint. The baseline therefore breaks its own ties on a seeded draw of a billionth of a dollar — too small to move MAE or RMSE by a cent, large enough to make the ranking explicitly random. What the capture column then reports is the honest floor: what a program would capture by choosing customers at random, which is the selection share itself.

Code 8.9. Express the clustering as a pipeline step

```
class IntensityClusters(BaseEstimator, TransformerMixin):
    """Behavioral clustering as a pipeline step.

    Inside a pipeline the scaler and the centroids are refitted on each
    training fold and only ASSIGN the customers being scored, which is
    what Table 8.2's split-contamination row requires of a learned
    feature. Clusters are relabeled by ascending revenue_per_month so the
    indicator columns are comparable across folds. That ordering rule is
    what makes them comparable; it does NOT make a given rank a named
    Chapter 6 segment, and the prose below says why it cannot."""

    def __init__(self, k=4, random_state=42):
```

```
        self.k = k
        self.random_state = random_state

    def fit(self, X, y=None):
        self.scaler_ = StandardScaler().fit(X)
        self.kmeans_ = KMeans(n_clusters=self.k, n_init=10,
                              random_state=self.random_state)
        self.kmeans_.fit(self.scaler_.transform(X))
        centers = pd.DataFrame(
            self.scaler_.inverse_transform(self.kmeans_.cluster_centers_),
            columns=list(X.columns))
        order = np.argsort(centers["revenue_per_month"].to_numpy())
        self.rank_ = np.empty(self.k, dtype=int)
        self.rank_[order] = np.arange(self.k)
        return self

    def transform(self, X):
        labels = self.rank_[
            self.kmeans_.predict(self.scaler_.transform(X))]
        # k - 1 indicators, lowest-intensity cluster as the reference.
        return np.column_stack([(labels == j).astype(float)
                                for j in range(1, self.k)])
```

Expected output: nothing — this cell defines one class.

Input: none. Transformation: one transformer definition. Output: a learned feature that a cross-validation routine refits per fold, assigning rather than learning from the customers it scores.

Its one non-obvious line is the relabeling, and it repays attention: k-means cluster numbers are arbitrary integers that change between fits, so an indicator column built from raw labels would mean a different thing in every fold — a bug that produces no error message and quietly destroys the candidate's estimate. Sorting clusters by ascending revenue per month is a declared labeling rule, and declaring it is what makes the columns comparable across folds.

It is worth being exact about what that ordering rule does not buy, because the temptation is to read these indicators as Chapter 6's segments returning. They are not. Chapter 6's four segments were named from full centroid profiles and validated against designed structure; these clusters are ranked on a single dimension, monthly revenue, and the second-lowest-intensity cluster in one fold can differ materially in discount behavior, channel mix, or basket shape from the second-lowest-intensity cluster in another. The rank is comparable; the customer type behind it is not guaranteed to be. Carrying the actual Chapter 6 segments into a predictive frame would require a different object: a frozen assignment rule, established before this study and applied unchanged to every fold, which is the third repair Section 8.9 listed. This lab takes the first, and names the candidate accordingly.

Code 8.10. Register the candidates and declare the selection rule

```
# Chapter 7's interpretable predictors, carried over unchanged.
EXPLAIN = ["recency_days", "tenure_days", "app_user",
           "email_opt_in", "discount_share"]
CLUSTER_INPUTS = ["recency_days", "frequency", "monetary",
                  "orders_per_month", "revenue_per_month", "aov",
                  "discount_share", "store_share"]
EXPLAIN_COLUMNS = sorted(set(EXPLAIN) | set(CLUSTER_INPUTS))

# A specification chosen for prediction: everything the snapshot
# legitimately knows, including the incumbent rule's own column.
PREDICT = ["spend_prior6m", "monetary", "frequency", "aov", "recency_days",
           "tenure_days", "orders_per_month", "revenue_per_month",
           "discount_share", "store_share", "occasionwear_share",
           "app_user", "email_opt_in"]

explanatory_style = Pipeline([
    ("features", ColumnTransformer([
```

```
        ("numeric", "passthrough", EXPLAIN),
        ("clusters", IntensityClusters(), CLUSTER_INPUTS)])),
    ("model", LinearRegression())])

prediction_oriented = make_pipeline(StandardScaler(), LinearRegression())

flexible = make_pipeline(StandardScaler(),
                         PolynomialFeatures(degree=2, include_bias=False),
                         LinearRegression())

INCUMBENT = "last-value baseline"
CANDIDATES = {
    "mean baseline":        (MeanBaseline(),      ["spend_prior6m"]),
    INCUMBENT:              (LastValueBaseline(), ["spend_prior6m"]),
    "explanatory-style + clusters": (explanatory_style, EXPLAIN_COLUMNS),
    "prediction-oriented":  (prediction_oriented, PREDICT),
    "prediction-oriented, degree 2": (flexible,   PREDICT),
}

# THE SELECTION RULE, declared here -- before a single candidate is
# fitted -- and executed unedited in Code 8.13.
MIN_CAPTURE_MARGIN = 0.02   # capture points a candidate must add
MIN_FOLDS_WON      = 4      # of five, on paired fold differences
SIMPLICITY_ORDER   = ["mean baseline", INCUMBENT, "prediction-oriented",
                      "explanatory-style + clusters",
                      "prediction-oriented, degree 2"]

# The cheapest leakage guard in the chapter: no candidate sees the label.
for name, (_, columns) in CANDIDATES.items():
    assert set(columns) <= set(train_df.columns), name
    assert "spend_next6m" not in columns, f"{name} was handed the label"
assert set(SIMPLICITY_ORDER) == set(CANDIDATES)
print("candidates registered:", len(CANDIDATES))
```

Expected output: the message that five candidates were registered, after two structural assertions confirm that every candidate's column list exists in the training frame and that none of them contains the label.

Input: the training frame's column names. Transformation: three pipelines and one registry. Output: a dictionary pairing each candidate with the columns it is allowed to see — which is also, read the other way, a written record of what each candidate is forbidden to see.

Note where the learned steps sit. The explanation-oriented candidate's segmentation is a step inside its pipeline, not a column computed beforehand; the two prediction-oriented candidates standardize inside their pipelines rather than on the table. Neither choice changes a single fitted value when a model is fitted once on everything. Both change the cross-validated estimate, because a pipeline is refitted per fold and a precomputed column is not.

The final assertion is the cheapest leakage guard in the chapter and the one most worth copying into every future project: no candidate's feature list may contain the label. It would not catch a subtle arithmetic relative, which is what the derivation trace is for, but it catches the version that actually happens under deadline — a column list built by selecting everything numeric.

Code 8.11. Build the training-only cross-validation leaderboard

```
from sklearn.metrics import mean_absolute_error, root_mean_squared_error
from sklearn.model_selection import (KFold, cross_val_predict,
                                     cross_validate)


def capture(y_true, y_pred, ids, share=None):
    """Share of actual outcome spend held by the customers a score ranks
    inside the program's capacity. One function, used by the folds, by
```

```
    the test set, and by the campaign list -- so all three apply the
    identical ranking rule, ties included."""
    share = SELECTION_SHARE if share is None else share
    ranked = (pd.DataFrame({"actual": np.asarray(y_true, dtype=float),
                            "pred": np.asarray(y_pred, dtype=float)},
                           index=ids)
              .rename_axis("customer_id").reset_index()
              .sort_values(["pred", "customer_id"],
                           ascending=[False, True]))
    k = max(1, round(len(ranked) * share))
    total = ranked["actual"].sum()
    if total <= 0:
        return np.nan
    return ranked.head(k)["actual"].sum() / total


def capture_scorer(estimator, X, y):
    """A scorer callable rather than a make_scorer wrapper: the
    tiebreak needs X's customer index, which a metric never sees."""
    return capture(y, estimator.predict(X), X.index)


SCORING = {"capture": capture_scorer,
           "mae": "neg_mean_absolute_error",
           "rmse": "neg_root_mean_squared_error"}

# One fold generator, built once, shared by every candidate: same split.
CV = KFold(n_splits=5, shuffle=True, random_state=42)

rows, fold_capture = [], {}
for name, (estimator, columns) in CANDIDATES.items():
    X = train_df[columns]
    scores = cross_validate(estimator, X, y_tr, cv=CV, scoring=SCORING)
    fold_capture[name] = scores["test_capture"]
    # Pooled out-of-fold predictions give ONE ranked list over all
    # training customers, which is the shape a campaign actually takes.
    oof = cross_val_predict(estimator, X, y_tr, cv=CV)
    rows.append({"candidate": name,
                 "cv_capture": scores["test_capture"].mean(),
                 "capture_sd": scores["test_capture"].std(),
                 "oof_capture": capture(y_tr, oof, X.index),
                 "cv_mae": -scores["test_mae"].mean(),
                 "cv_rmse": -scores["test_rmse"].mean()})

leaderboard = pd.DataFrame(rows).set_index("candidate")
print(leaderboard.round(4).to_string())
```

Expected output: one table with five rows and seven columns: cross-validated capture with its fold-to-fold standard deviation, cross-validated MAE with its standard deviation, cross-validated RMSE, and each candidate's capture and MAE margins over the incumbent.

Input: the training customers only. Transformation: five-fold cross-validation of every candidate on one shared fold generator, scored on all three declared metrics. Output: the leaderboard, and the only evidence the analyst is permitted to use when choosing a finalist.

The custom scorer is where Section 8.6's argument becomes executable, and its shape is deliberate. Capture is not a built-in metric because it is not a property of the errors; it is a property of the ranking, computed by sorting the fold's customers by predicted score, taking as many as the capacity share funds, and asking what share of that fold's actual outcome spend those customers held. That is why it is written as a scorer callable taking the estimator, the features, and the labels, rather than as a metric function wrapped by make_scorer: a metric function receives two arrays of numbers and never sees who the customers are, so it cannot break ties on customer identity. The campaign

list does break ties on identity, and a leaderboard that ranks by a different rule than the list it is choosing has quietly changed the question. One capture function, used by the folds, by the test set, and by the final list, is the fix.

Two capture columns appear, and they answer different questions. The cross-validated figure averages five fold-level rankings: it estimates how well the model orders a population of that size, and its spread across folds is the reliability signal. The out-of-fold figure pools one held-out prediction per training customer into a single ranked list, which is the shape a campaign actually takes. For a metric that cannot be decomposed customer by customer — and capture, being a property of a cutoff, cannot be — the two need not agree, and a wide gap between them is itself informative about how much the ranking depends on the size of the pool being ranked.

The margin columns are the deliverable's sentences in numerical form — a model that beats the mean baseline handsomely and the incumbent barely has learned something and earned little, and only the second comparison is a budget argument. The per-fold scores are stored rather than discarded, because Code 8.13 needs them.

Read the standard-deviation columns as seriously as the means, because they are the variance half of Section 8.8 made visible. A candidate whose capture swings widely across folds is telling you its performance depends on which whales it drew, and a margin smaller than the fold-to-fold spread of the metric it is measured on is not yet a finding. The designed structure of the base fixes the pattern your predictions should anticipate: the mean baseline's capture should land near a tenth, since a constant score produces an arbitrary ranking; the incumbent should beat it comfortably, because trailing spend genuinely carries signal; and the models should beat the incumbent by a margin that is real and considerably smaller than anyone hoped, because most of what makes a customer valuable next season is visible in what they did last season.

VERIFICATION CHECK

Before you run: write four predictions. Which candidate wins on capture, and which on MAE — and state explicitly what you will do if they disagree. How far will the mean baseline's capture sit from one tenth, and why can it not sit far from it? Will the explanatory-style candidate beat the prediction-oriented one, and what would it mean about Chapter 7's variable selection if it did? Will the degree-2 candidate beat its linear parent, and by more than the fold-to-fold spread?

After you run: grade every prediction, then run the three comparison clauses of Section 8.9 as an audit of your own table. Same split: confirm that one fold generator produced every row — the code shares one CV object, and the point of reading the cell is to see that it does. Same metric: the finalist will be chosen on the declared decision metric by the rule in Code 8.13, and if a candidate wins on capture while losing on MAE, that is a real finding to report rather than a licence to switch metrics. Write the sentence it forces: this candidate orders customers better and prices them worse, which is acceptable for a fixed-capacity list and unacceptable if anyone later uses the score as a dollar estimate — so the permitted-use note of Section 8.15 acquires a clause. Same baseline: state each model's margin over the incumbent in both metrics, and say whether it exceeds the fold-to-fold spread.

Investigate if: a baseline wins. That is a legitimate outcome, not a failed lab, and it has a specific meaning: on this frame, with these features, at this horizon, the sorted spreadsheet is the honest answer, and the deliverable should say so and redirect the effort to framing — a shorter horizon, a different target, or a treatment question an experiment could settle. Investigate also if any model's capture exceeds roughly half, or its MAE falls to a small fraction of the incumbent's: against a base where a quarter of customers spend nothing and spend is concentrated in a small minority, that is the too-good signature of Section 8.4, and the feature trace comes before the celebration.

8.12.4 Lab 8.2, Part B: Manufacturing Overfitting Inside the Training Data

Part B makes the U-shape of Section 8.8 visible on StyleCraft's own data by turning the one dial this chapter's machinery offers: flexibility via polynomial features. The design of this demonstration is itself a lesson, because the

obvious version of it violates the chapter's own rule. Fitting nine degrees and reading the test error at each one would let the test set choose the model's complexity — development steered by the sealed data, the exact practice Section 8.5 forbids and the practice that makes a final test figure meaningless. The dial is therefore turned entirely inside the training customers, with cross-validation supplying the held-out estimate at every degree, and the test set stays sealed until Part C names a finalist and opens it.

The second design choice is scale. Expanding all thirteen prediction-oriented features to the ninth degree would generate an unreadable number of terms and demonstrate mostly that a laptop can run out of patience. Two predictors — recency and tenure, both continuous, both genuinely related to future spending — expand to a few dozen terms at the top degree, which is enough to overfit visibly and few enough to interpret.

Code 8.12. Demonstrate overfitting inside the training data

```
# Two predictors, not the whole table. With eight inputs, degree 9 expands
# to more than twenty-four thousand terms before the regression is fitted;
# with two, it expands to fifty-four, which is a demonstration rather than
# a computation (scikit-learn developers, 2026c).
POLY_FEATURES = ["recency_days", "tenure_days"]

rows = []
for degree in range(1, 10):
    pipe = make_pipeline(
        StandardScaler(),
        PolynomialFeatures(degree=degree, include_bias=False),
        LinearRegression())
    scores = cross_validate(pipe, train_df[POLY_FEATURES], y_tr, cv=CV,
                            scoring=SCORING, return_train_score=True)
    pipe.fit(train_df[POLY_FEATURES], y_tr)
    rows.append({
        "degree": degree,
        "terms": pipe.named_steps["polynomialfeatures"].n_output_features_,
        "train_rmse": -scores["train_rmse"].mean(),
        "train_mae": -scores["train_mae"].mean(),
        "cv_mae": -scores["test_mae"].mean(),
        "cv_mae_sd": scores["test_mae"].std(),
        "cv_capture": scores["test_capture"].mean(),
    })

flex = pd.DataFrame(rows).set_index("degree")
print(flex.round(3).to_string())

# Least squares minimizes SQUARED error, so training RMSE cannot rise as
# nested terms are added. Training MAE carries no such guarantee.
assert (flex["train_rmse"].diff().dropna() <= 1e-6).all()
mae_rises = int((flex["train_mae"].diff().dropna() > 0).sum())
print(f"\ndegrees at which training MAE rose anyway: {mae_rises}")
print(f"best cross-validated degree on MAE: {flex['cv_mae'].idxmin()}")
print(f"worst cross-validated degree on MAE: {flex['cv_mae'].idxmax()}")
```

Expected output: a nine-row table carrying, per degree, the number of polynomial terms, the training RMSE and MAE, the cross-validated MAE with its fold-to-fold spread, and the cross-validated capture; then an assertion on the monotone training RMSE, a count of degrees at which training MAE rose anyway, and the best and worst cross-validated degrees.

Input: two training-set predictors. Transformation: nine pipelines of increasing flexibility, each cross-validated on the shared folds. Output: the two curves of Section 8.8, printed side by side, with the test set untouched.

Read the columns in the order they were argued. The term count rises steeply, which is the whole meaning of flexibility. Training RMSE falls, or at worst holds, at every step, and the assertion says why that is guaranteed rather than observed: least squares minimizes squared error, and a richer nested model can always reproduce a poorer one's

fit. Training MAE is printed beside it precisely because it carries no such guarantee — the fitting criterion protects squared error, not absolute error, so a richer model can trade a few cents of typical miss for a large reduction in a handful of squared ones. The cell counts the degrees at which that happens rather than asserting it cannot.

The cross-validated column is the one that decides anything, and it should be read as a pattern to investigate rather than a shape guaranteed to appear. The expected behavior is a fall, a turn, and a climb, with the turn marking the honest amount of flexibility for this sample size. What is not guaranteed is a clean U in one finite sample, or that the highest degree is the worst performer: fold-to-fold noise is real, the printed spread column measures it, and a difference smaller than that spread is not a difference. Name the two sides with Section 8.8's vocabulary, locate the turn, and state whether the distance between the best and worst degrees exceeds the fold-to-fold noise.

VERIFICATION CHECK

Before you run: predict the three columns' shapes. Training RMSE: state the direction and say whether it is guaranteed or merely expected, and why. Training MAE: same two questions, and a different answer. Cross-validated MAE: predict where the turn falls, and write down what you will conclude if there is no visible turn at all within nine degrees.

After you run: check the guaranteed claim against the assertion, the unguaranteed one against the printed count, and the U-shape against the fold-to-fold spread. Then answer the leakage question this cell exists to pose, which the draft of this chapter got wrong and which is worth getting right in writing. Suppose you had expanded the polynomial features on the pooled data before splitting. Would that be split contamination? The answer is no, and the reason is the estimation test of Section 8.4: PolynomialFeatures computes predetermined powers and products, so fitting it mainly establishes how many output columns there will be and what to call them; it estimates nothing from the values, and expanding before or after the split produces identical numbers for every row (scikit-learn developers, 2026c). Now ask the same question about the first step in the same pipeline. StandardScaler estimates a mean and a standard deviation from the data it is fitted to, so fitting it on the pooled table before splitting would let every validation customer contribute to the centering and scaling applied to the training customers — split contamination, Table 8.2's fourth row, committed by the blandest step in the pipeline. Write both answers down with the reason attached, because the distinction between a learned step and a fixed transformation is what the rule actually is, and “do everything after the split” is the rule of thumb that people remember and then apply to the wrong things.

Investigate if: cross-validated MAE improves monotonically across all nine degrees. That is not the designed result and usually means the flexibility dial is not being turned far enough to hurt on this sample size — eight thousand customers tolerate more flexibility than eight hundred, exactly as Section 8.8 said. Extend the range, or add a third predictor, and report what you changed.

8.12.5 Lab 8.2, Part C: The Selection Rule and the Sealed Test Set

Everything so far happened inside the training customers. Part C ends development by executing the selection rule, and only then spends the test set — once, on the candidate the rule names.

The rule matters because the obvious alternative is wrong in a way that is easy to miss. Taking whichever candidate posts the highest cross-validated capture would name a winner even when its advantage over the incumbent is a fraction of a percentage point, even when the advantage appears in two folds out of five, and even when a materially simpler candidate performs indistinguishably. That is the leaderboard being read as a scoreboard rather than as evidence, and it contradicts this chapter's own warning that a margin smaller than the fold-to-fold spread is not yet a finding.

So the rule is written into Code 8.10, before any candidate is fitted, and Code 8.13 executes it unedited. It has four clauses: a candidate must add at least a stated number of capture points over the incumbent; it must win at least four of the five folds; among the candidates that clear both, the simplest wins, on a complexity ordering declared in the same cell; and if nothing clears the bar, the incumbent is retained. The threshold is a business judgment rather than a statistical one — two capture points on this program is roughly the margin at which the model's list is worth the operational change of adopting it — and writing it down in advance is what stops it from becoming whatever the results happen to justify.

The comparison is paired, and the reason is the one Section 8.9 gave. Folds differ in difficulty: a fold holding three whales is hard for every candidate at once. Comparing a candidate's mean capture against its own standard deviation across folds therefore measures mostly fold difficulty, which every candidate shares and which the comparison should cancel. Subtracting the incumbent's score fold by fold cancels it, and the spread of those differences is the honest measure of whether the candidate reliably wins or merely averages well.

Code 8.13. Apply the declared selection rule

```
# The rule declared in Code 8.10, executed unedited. Because every
# candidate ran on the same folds, the informative quantity is the PAIRED
# per-fold difference against the incumbent, not each candidate's own
# standard deviation: a fold that is hard for one model is usually hard
# for all of them, and pairing removes that shared difficulty.
base = fold_capture[INCUMBENT]

rows = []
for name, scores in fold_capture.items():
    if name == INCUMBENT:
        continue
    diff = scores - base
    clears = bool(diff.mean() >= MIN_CAPTURE_MARGIN
                  and (diff > 0).sum() >= MIN_FOLDS_WON)
    rows.append({"candidate": name,
                 "mean paired margin": diff.mean(),
                 "sd of paired margins": diff.std(),
                 "folds won": int((diff > 0).sum()),
                 "clears the rule": clears})

paired = pd.DataFrame(rows).set_index("candidate")
print(paired.round(4).to_string())

qualified = [n for n in SIMPLICITY_ORDER
             if n in paired.index and paired.loc[n, "clears the rule"]]
FINALIST = qualified[0] if qualified else INCUMBENT

print(f"\nrule: at least {MIN_CAPTURE_MARGIN:.2f} capture over the "
      f"incumbent and at least {MIN_FOLDS_WON} of 5 folds won")
print("qualifying, simplest first:", qualified or "none")
print("FINALIST:", FINALIST)
```

Expected output: a four-row table of paired margins — mean, spread, folds won, and whether the rule is cleared — then the rule restated, the qualifying candidates in simplicity order, and the finalist.

Input: the per-fold capture scores stored by Code 8.11. Transformation: subtraction, fold by fold, against the incumbent. Output: a named finalist, arrived at by a rule rather than by a preference.

Read the paired-margin table against the leaderboard and notice how differently the two read. On the leaderboard, a candidate's capture standard deviation is often larger than its margin over the incumbent, which invites the conclusion that nothing is distinguishable. The paired column usually tells the opposite story, and correctly: once shared fold difficulty is subtracted out, a candidate that wins every fold by a consistent amount is showing exactly the reliability the raw spread concealed. This is the same reasoning that makes a paired comparison

stronger than two independent ones anywhere else in statistics, and it is available here only because the frame insisted on one fold generator for every candidate.

Note also what the simplicity clause does when several candidates qualify. It selects the simplest, not the highest — so a flexible candidate that edges out its linear parent by less than the declared margin does not win, and the deliverable carries the model whose behavior is easiest to explain, monitor, and retrain. Section 8.8's bias–variance argument said flexibility must earn its place; the rule is what makes that sentence enforceable. And if no candidate clears the bar, the finalist is the incumbent, which is a legitimate and reportable outcome rather than a failed lab.

Code 8.14. Open the sealed test set, once

```
# The finalist was named by Code 8.12 from training folds alone. Refit it
# on ALL training customers -- the cross-validated estimate came from
# models fitted on four folds at a time -- and open the test set once.
final_model, final_columns = CANDIDATES[FINALIST]
final_model.fit(train_df[final_columns], y_tr)

pred_model = final_model.predict(test_df[final_columns])
pred_mean  = np.repeat(y_tr.mean(), len(y_te))   # trained mean, not test
pred_lv    = test_df["spend_prior6m"].to_numpy()

results = []
for name, pred in [("mean baseline", pred_mean),
                   (f"{INCUMBENT} (incumbent)", pred_lv),
                   (f"selected: {FINALIST}", pred_model)]:
    results.append({"rule": name,
                    "capture": capture(y_te, pred, test_df.index),
                    "MAE": mean_absolute_error(y_te, pred),
                    "RMSE": root_mean_squared_error(y_te, pred)})
print(pd.DataFrame(results).set_index("rule").round(3).to_string())

# NOW the halves may be compared: with the grade already recorded, this
# is context for reading it rather than knowledge that could steer it.
halves = pd.DataFrame(
    {"train": [y_tr.eq(0).mean(), y_tr.mean(), y_tr.quantile(0.99)],
     "test":  [y_te.eq(0).mean(), y_te.mean(), y_te.quantile(0.99)]},
    index=["zero-label share", "mean", "99th percentile"])
print("\n" + halves.round(3).to_string())

# An unconstrained linear model can predict negative spend. Count it in
# the open; do not pick a remedy by watching this number move.
negatives = int((pred_model < 0).sum())
print(f"\nnegative test predictions: {negatives} of {len(pred_model)} "
      f"({negatives / len(pred_model):.1%})")
```

Expected output: the name of the selected candidate, then a three-row table giving capture, MAE, and RMSE for the mean baseline, the incumbent, and the model on the same held-out customers, and finally the count and size of any negative predictions.

Input: the training customers, the sealed test customers, and the leaderboard's verdict. Transformation: one refit and three sets of predictions. Output: the deliverable's central table.

Two details enforce disciplines the chapter argued for. The finalist arrives as a variable set by Code 8.13, not as a name typed after seeing test results — which makes the order of operations auditable in the notebook itself rather than asserted in a methods paragraph. And the mean baseline's prediction is the training mean repeated, not the test mean: computing it from the test labels would hand the baseline a summary of the answers and turn the model's opponent into a cheat. That is the smallest possible instance of Section 8.5's rule that anything learned from data is learned from training data, and it is worth pointing at because it is the version students most often get wrong while getting everything larger right.

The half-comparison table appears here rather than at the split, and its position is the argument. Read before selection, those numbers steer development; read after the grade is recorded, they explain it — a test half holding fewer whales makes a strong capture figure slightly less impressive, and one holding more makes a weak MAE slightly more forgivable. Same numbers, different epistemic status, decided entirely by when they were looked at.

The negative-prediction line is an honest disclosure rather than a repair. An unconstrained linear model can return a negative predicted spend for a customer whose features sit at the low end of every input, and the number is nonsense as a dollar estimate even when the ranking it implies is perfectly sensible. Four remedies exist: keep them and document that the score is used only for ranking, which is what this program does; clip at zero; model a transformed response; or use a model family that cannot go negative. Three of the four are modeling choices, and the rule that governs them is the rule that governed everything else — a remedy is chosen by cross-validation on the training customers, never by trying each one against the test set and keeping whichever looked best.

VERIFICATION CHECK

Before you run: write four predictions. Will the model beat both baselines on capture, and on MAE? How close will the test figures sit to the cross-validated ones, and in which direction do you expect them to differ? What would an alarming result look like in each direction — too poor, and too good? And what will you do if the test capture disagrees materially with the cross-validated capture?

After you run: place the result on Table 8.4's signature grid and write the two sentences that survive into Section 8.13's deliverable: the capture sentence, which prices the model in the decision's own currency, and the margin sentence in dollars, which prices its typical miss against the incumbent's. Then honor the protocol: the test set is now spent. If the result prompts a new idea — another feature, another family, a clipping rule — that idea goes back to the training customers and the cross-validated leaderboard, and any future test figure is reported as what it would then be, a second look rather than a clean grade.

Investigate if: the test figures sit far better than the cross-validated ones. A modest gap in either direction is ordinary sampling variation, and the fold-to-fold spread from Code 8.10 is the yardstick for what counts as modest. A large gap in the model's favor is the too-good signature again, and the most common cause is not exotic: a column that entered the feature list after the leaderboard was frozen, or a test frame rebuilt at some point with a later snapshot. Both are found by rerunning Code 8.4 and Code 8.5 from the top and confirming that the row counts and the recency minimum are unchanged.

8.12.6 Lab 8.2, Part D: The Deliberate Leak, Caught

Part D commits this chapter's canonical error on purpose, because a leak you have built is a leak you will recognize. Rebuild the modeling table the lazy way — the way an unframed prompt or a hurried analyst would — by pointing the same feature builder at the June 30, 2026 analysis date, so that every column is computed across the full twenty-four months, outcome window included. Nothing else changes: same customers, same labels, same split, same features, same estimator. One argument moves, and the model's measured performance improves dramatically. That is the entire anatomy of the error, and the lab works to keep it that clean: the honest side of the comparison is a fresh copy of one fixed specification rather than whichever candidate the selection rule happened to name, because a demonstration that changed the model family and the feature table at the same time would prove nothing about leakage.

Code 8.15. Manufacture and detect temporal leakage

```
from sklearn.base import clone

# The lazy build: the June 30, 2026 feature table, whose every column was
```

```
# computed across the full twenty-four months -- outcome window included.
cf_full = build_customer_features(transactions_clean, customers, products,
                                  snapshot=pd.Timestamp("2026-06-30"),
                                  window_start=FEATURE_START)

leaky_df = (cf_full.loc[model_df.index]
            .join(labels).join(prior6)
            .fillna({"spend_next6m": 0.0, "spend_prior6m": 0.0}))

# The SAME CUSTOMERS, not merely the same seed. A seed reproduces
# positions, not identities, so a re-split could grade the two pipelines
# on different people -- which Section 8.9's same-split clause forbids.
lk_train, lk_test = leaky_df.loc[train_ids], leaky_df.loc[test_ids]
assert lk_train.index.equals(train_df.index)
assert lk_test.index.equals(test_df.index)
assert np.allclose(lk_test["spend_next6m"], y_te)

# ONE specification, fitted twice. clone() returns an unfitted estimator
# with identical parameters, so the only difference between these two
# models is which feature table they were shown. Using the selected
# finalist here would confound leakage with a change of model family.
honest_demo = clone(prediction_oriented).fit(train_df[PREDICT], y_tr)
leaky_demo  = clone(prediction_oriented).fit(lk_train[PREDICT],
                                             lk_train["spend_next6m"])

for tag, pred in [("honest", honest_demo.predict(test_df[PREDICT])),
                  ("leaky ", leaky_demo.predict(lk_test[PREDICT]))]:
    print(f"{tag} model  test MAE: "
          f"{mean_absolute_error(y_te, pred):8.2f}"
          f"   capture: {capture(y_te, pred, test_df.index):6.1%}")

# The audit line that convicts the column with no modeling at all: at a
# December 31 snapshot, a correct recency cannot be below 181 days.
impossible = int((cf_full["recency_days"] < 181).sum())
print(f"\nrange check -- customers whose recency is impossible at the "
      f"declared snapshot: {impossible:,} of {len(cf_full):,}")
```

Expected output: the honest and leaky models' test MAE and capture on the same customers, with the leaky figures decisively better, then the count of customers whose recency is arithmetically impossible at the declared snapshot; three assertions pass silently.

Input: the certified files, the existing labels, and the frozen train and test identities. Transformation: one feature table built at the wrong date, one model fitted on it. Output: the sensation of a model that is too good, produced on demand, and the audit line that convicts it.

clone() is the instrument that holds the specification fixed. It returns a new, unfitted estimator carrying identical parameters, so the honest and the leaky model differ in exactly one respect: which feature table they were shown. Using the selected finalist for the honest side would have been the natural shortcut and a confounded comparison — had the rule named a baseline, the cell would have been contrasting a last-value rule with a leaky regression and attributing the whole gap to leakage.

The three assertions are the cell's methodological spine, and the first two answer a question the draft of this chapter left open. Re-splitting the leaky frame with the same random seed would not guarantee the same customers on each side, because a seed reproduces positions rather than identities and any difference in row order sends different people to the test half. Selecting by the stored index does guarantee it, and the assertions prove it — after which the two pipelines differ in exactly one respect and the comparison means what it claims. The third assertion confirms that the labels themselves were not disturbed by the rebuild, which is what makes the two MAE figures comparable at all.

State the mechanism in writing before reading the numbers. The full-window table's recency was measured to June 2026, so a customer who bought in May has a small recency that is only possible because of an outcome-

window order; its monetary sums feature-window spend plus the label itself, which is Table 8.2's arithmetic-leakage row; its frequency and its per-month rates absorb outcome-window orders the same way. The model is not predicting the future. It is reading it, and reporting how well the future predicts itself.

Then catch it the way an auditor would who had not built it. The range check is the instrument, and it needs no model at all: at a December 31, 2025 snapshot, a correctly computed recency cannot be smaller than 181 days, because a smaller value would require a purchase after the snapshot. The printed count of impossible values is the smoking gun — thousands of them, produced by a single wrong argument. Close with the sentence the whole chapter has been building toward, and keep it for the vendor meeting: this leaky model's excellent score is exactly what a near-perfect accuracy claim looks like from the inside, and note that it would survive a train/test split, a fixed seed, a cross-validated leaderboard, and a tidy comparison table, because the leak sits in the features, where no split can see it. Splits catch overfitting; only audits catch leaks.

8.12.7 Lab 8.2, Part E: The List, the Money, and the AI Round Trip

Part E translates the surviving model into the decision. Score the test customers with the honest model, rank, and cut the top decile; do the same with the incumbent's trailing-spend ranking; then compare what each list captures of the test customers' actual outcome-window spend, and count how many names actually change.

Code 8.16. Compare the two ranked lists

```
scored = test_df.assign(pred=pred_model).reset_index()
assert "customer_id" in scored.columns

# The same capacity share, the same tiebreak, the same function the folds
# used -- so the number in the deliverable is the number that was raced.
k = max(1, round(len(scored) * SELECTION_SHARE))
top_model = scored.sort_values(["pred", "customer_id"],
                               ascending=[False, True]).head(k)
top_inc = scored.sort_values(["spend_prior6m", "customer_id"],
                             ascending=[False, True]).head(k)

total = scored["spend_next6m"].sum()
assert total > 0
assert len(top_model) == len(top_inc) == k

cap_model = top_model["spend_next6m"].sum() / total
cap_inc = top_inc["spend_next6m"].sum() / total
overlap = len(set(top_model["customer_id"]) & set(top_inc["customer_id"]))
assert np.isclose(cap_model, capture(scored["spend_next6m"],
                                     scored["pred"],
                                     scored["customer_id"]))

print(f"selection share {SELECTION_SHARE:.1%} -> {k} slots on this "
      f"test population ({CAPACITY} in deployment)")
print(f"model list captures:     {cap_model:6.1%}")
print(f"incumbent list captures: {cap_inc:6.1%}")
print(f"capture gap:             {cap_model - cap_inc:+6.1%}"
      f"  =  ${(cap_model - cap_inc) * total:,.2f} of outcome spend")
print(f"customers on both lists: {overlap} of {k}"
      f"  ({k - overlap} names change)")
```

Expected output: the number of slots on the test population, the capture of each list, the capture gap expressed both in percentage points and in dollars, and the overlap count with the number of names that change; three assertions pass silently.

Input: the sealed test customers and their scores. Transformation: two deterministic sorts and one arithmetic comparison. Output: the exhibit the meeting is actually deciding on.

The cutoff is the selection share computed in Code 8.6, not a typed tenth, so the number of slots on the test population is the program's capacity scaled to the population being ranked. The final assertion re-derives the same figure through the capture function the folds used, which is a small proof that the leaderboard and the campaign list are measuring one thing rather than two things with one name.

The sorts carry a declared secondary key, and the reason is operational rather than statistical. Predicted values tie, trailing-spend values tie far more often — every customer with no purchases in the trailing window ties at zero — and a sort with no tiebreaker returns whichever order the frame happened to be in. The eight hundredth name would then differ between the analysis notebook and the campaign build, which is the kind of discrepancy that destroys trust in a program for reasons no one can reconstruct a month later. Sorting by score and then by customer identifier makes the list reproducible by anyone, on any machine, in any row order.

Read the three outputs as three different arguments. The capture gap in percentage points is the model's ordering quality. The same gap in dollars is the budget argument, and it is the number the deliverable leads with. The overlap count is the change-management argument, and it is the one the CRM manager will ask about first: a model that reorders the list without changing many names is buying a small improvement at the cost of an operational change, while a model that replaces a large fraction of the list is making a substantial claim about who the program has been missing — and the customers it adds should be recognizably the lumpy occasion profile the miniature predicted.

Close the lab with two exercises in judgment. First, the expected-value sketch of Section 8.10, one paragraph of arithmetic: at twenty dollars per slot, what would the capture gap have to be worth in incremental margin for the model to out-earn the spreadsheet — and underline the word incremental, with two sentences on why this lab cannot certify it. The model predicts spend, not spend caused by gifting; the capture gap says that the model's eight hundred customers spent more than the spreadsheet's eight hundred, and says nothing about how much either group would have spent untreated. Section 7.13's discipline names the error and Chapter 11's instrument is the only thing that closes it.

Second, the AI round trip. Hand an assistant the raw certified tables and the deliberately loose prompt “build me the best possible model to predict customer spend — maximize accuracy,” then run Table 8.5's five points on what comes back, in order, in writing. The designed likelihood — engineered by the loose prompt itself — is a pipeline that fails point 1 (no frame, no snapshot, no eligibility rule), point 2 (full-window features, exactly Code 8.13's leak, arrived at innocently), point 3 (complexity chosen against a test split, preprocessing fitted on pooled data), and point 4 (no baseline), while reporting a fit statistic the assistant celebrates. Your audit memo, filed per Appendix D, is the lab's final deliverable and the direct rehearsal of Section 8.13's vendor verdict. Deliverables for both labs: the notebook with all fourteen cells executed, the written predictions made before each cell was run, the reconciled population counts from Code 8.5, the frozen leaderboard from Code 8.10, the flexibility table from Code 8.11, the single test-set table from Code 8.12, the leakage audit from Code 8.13 with its range-check count, the list comparison from Code 8.14, and the AI-Use Appendix documenting every assistant exchange, including at least one delegated step you corrected and why.

8.13 Marketing Interpretation and Managerial Insight

The lab's outputs are leaderboards and ranked lists; the thirty-day deadline runs on sentences. This section translates — and, per this guide's standing practice, it does so partly by exhibiting wrong managerial readings and correcting them, because the predictive setting mints new misreadings faster than the associational one did, and three of them will be voiced in the scoring meeting by people whose only error is applying last chapter's intuitions to this chapter's numbers.

The first wrong reading will come from whoever liked the vendor deck: “Our in-house model's average miss is fifty-odd dollars. The vendor's model is at 96 percent. Ours is a C-minus student; theirs is an A. Buy the license.” The comparison is meaningless before it is even unfair, and the first problem is vocabulary rather than evidence. “Ninety-six percent accuracy” is undefined for a numeric prediction. Accuracy is a classification statistic — the share of cases assigned to the correct class — and future spend in dollars has no classes to be correct about. If the vendor means a test R^2 of 0.96, they should say so and disclose the evaluation design, because R^2 is a proportion of variance accounted for and not a percentage of predictions that were right; the two sentences sound alike and

describe different quantities. It is worth knowing, and worth saying in the meeting, that R^2 computed on data a model was not fitted to is not even bounded below by zero: a model that predicts held-out cases worse than their own mean produces a negative value (scikit-learn developers, 2026d). A statistic that can go negative out of sample is not a percentage of anything.

The second problem is the one Section 8.4 armed you for. Even read charitably as $R^2 = 0.96$ on individual six-month spend, the number is implausible against everything the lab documented about this outcome: a quarter of eligible customers spend nothing, spend is concentrated in a small minority, and the honest baselines land in a known range. The corrected reading, which the deliverable should print in exactly this shape: “Our model's grade was earned under exam conditions we can show you — a declared frame, a sealed test set opened once, two baselines on the same customers. The vendor's number, as presented, was not earned under any conditions we can inspect, and the statistic it names does not exist for this kind of target. Before the two numbers can share a sentence, the vendor must answer five questions” — and the five questions are Table 8.5 read aloud: What is the frame, and what exactly does the number measure? Where is the snapshot, and can you prove no feature crosses it? What was held out, how often was it consulted, and was model complexity chosen against it? What baseline was beaten, by how much, on which metric? What is the error in dollars, on our customers, and what happens to it over time? A vendor with good answers becomes a genuine candidate — build-versus-buy is a real decision, and Section 8.14's third vignette shows the buy side winning honestly. A vendor without them is selling in-sample flattery at an annual license price.

The second wrong reading is gentler and more corrosive: “The model says C004471 will spend 585 dollars, so put her on the list — the model knows.” The model does not know. The score is a conditional average — what customers with C004471's snapshot profile spent, on average, in past outcome windows — wearing the precision of its decimal places, and the test MAE is the printed reminder that individual customers sit tens to hundreds of dollars from their scores routinely. The corrected reading changes the object of confidence from the customer to the list: the model is not trusted to be right about C004471; it is trusted to sort eight thousand customers well enough that the top eight hundred capture more of next season's spend than any competing list — which is precisely the claim the capture exhibit tests and the reason Section 8.6 made capture the decision metric. This distinction — score as verdict versus score as ordering — is the predictive sibling of Chapter 7's coefficient-is-not-a-lever box, and it changes what “the model was wrong” even means: one customer's shortfall is noise; the list underperforming the spreadsheet is failure.

The third wrong reading is the most expensive, and it is the one the opening brief pre-empted on purpose: “The model's list captures six points more spend than the spreadsheet's, so the Backstage program will generate that much additional revenue.” It will not, and the gap between those two sentences is the difference between a prediction and a treatment effect. The capture figure says that the customers the model selected turned out to spend more than the customers the spreadsheet selected. It says nothing about how much either group would have spent without a lookbook. A program that gifts eight hundred customers who were always going to spend heavily has bought recognition, not growth, and the honest deliverable prices it that way: the model improves who receives the treatment, and only a holdout — a randomly withheld slice of the eligible list, measured six months later — can price what the treatment does. Naming that experiment in the deliverable is not a hedge. It is the analysis pricing its own upgrade, the same closing move Chapter 7's drivers deliverable made.

The deliverable that survives all three misreadings has a fixed anatomy, and the lab produced every part: the frame, one page, seven declarations and the label definition with its four dates, because it is the document every later question lands on; the leaderboard from Code 8.11, cross-validated on training customers only, with both baselines and every candidate's margin; the single test-set table from Code 8.14, with capture leading and MAE and RMSE in dollars beside it; the capture head-to-head from Code 8.16, model list against spreadsheet list, with the dollar gap and the count of names that change, because the meeting is deciding between those two artifacts and this table grades the actual choice; the expected-value sketch, with incremental flagged as the assumption the design cannot certify and the holdout test named as the instrument that could; the vendor verdict as a one-paragraph memo built on the five questions; and the monitoring plan of Section 8.10 — who checks the realized error when the outcome window matures in June, against what benchmark, which input distributions are watched in the meantime, and what

number triggers a retrain — because the deliverable is not a scored list but a scored list with a maintenance contract, and only one of those survives its first drift.

8.14 Business Analytics in Practice

This section turns from the fictional case to how predictive models operate — and fail — in professional marketing organizations, where the models are commodities, the pipelines are products, and the failures that matter are almost never statistical. All three vignettes below are composites of recurring professional patterns rather than reports about a single named organization.

8.14.1 Lead Scoring and the Adoption Problem

The first vignette is lead scoring, the business-to-business world's version of this chapter's scored list. The setting recurs across enterprise software firms: marketing builds a propensity model scoring inbound leads on firmographic fit and behavioral signals, validated properly out of sample, demonstrably better than the alphabetical-and-instinct routing it replaced — and six months later, sales representatives are ignoring the scores. The post-mortems rarely find a modeling failure. They find an adoption failure with a recurring anatomy: nobody translated the score into the representative's decision, so a lead marked 74 arrived with no instruction about what to do before lunch; nobody showed the representatives evidence that the ordering worked, so the first false positive — a high-scored lead who wasted an afternoon — confirmed every prior about ivory-tower analytics; and nobody instrumented the loop, so scores were computed, exported, and quietly unconsulted while the dashboard reported the model live.

The operational artifact that separates the teams who fix this from the teams who repeat it is unglamorous: an adoption metric, defined as precisely as any model metric and reported beside it. The working version is a contact-rate-by-score-decile report — of the leads the model placed in the top decile, what share were actually worked within the service-level window, and how does that share compare with the bottom decile? A model whose top and bottom deciles are worked at the same rate is not in production in any sense that matters, whatever the pipeline logs say. Teams that instrument this discover the gap in weeks rather than quarters, and the fix is usually not a better model but a translation layer: score bands with named actions rather than a continuous number, a weekly list rather than a field in a record, and a published back-test showing representatives what the ordering was worth on leads they remember. The practice lesson is the one Provost and Fawcett (2013) build their expected-value framing around: the model is an input to a decision system, and the system — thresholds in the user's units, feedback that earns trust, telemetry on whether scores change behavior — is what produces value. A perfectly ranked list that no one acts on has an expected value of exactly zero.

8.14.2 Silent Decay After a Pricing Change

The second vignette is silent decay, composited from a pattern retention teams describe often enough to treat as a genre. A subscription commerce firm builds a customer-value model — a spend-horizon frame much like this chapter's — and wires it into onboarding offers, service-tier routing, and paid-acquisition bidding. It works, and because it works, it stops being watched. Fourteen months later, someone notices acquisition economics sagging and pulls the thread: the model has been mis-scoring new customers for two quarters, ever since a pricing-and-packaging change restructured early purchase behavior — the model's most important input. Nothing had errored. The scores still arrived, plausible-looking, on schedule; the model was simply answering a question about a relationship that no longer held.

Note which failure this is, in the vocabulary of Section 8.10. The inputs did shift, so there was data drift. But the expensive part was concept drift: the same early-purchase pattern now implied a different lifetime value, because the packaging change altered what an early purchase meant (Gama et al., 2014). No amount of watching the input distributions alone would have caught the second, which is why the repaired practice monitors both. The operational artifact here is a monitoring view with three panels and one rule. The panels: realized error on each cohort as its outcome window matures, plotted against the benchmark the original test set established; the distribution of the two or three most important inputs, this month against the training window; and the score distribution itself, which moves when either of the first two does. The rule is the part teams skip: a written trigger naming the number that

forces a retrain — realized error exceeding the test benchmark by a stated margin for two consecutive cohorts — plus a standing clause that known structural changes, a repricing or a merchandising overhaul, force revalidation without waiting for the cadence. Sculley et al. (2015) systematized why this is engineering rather than diligence: the model is the smallest box in the system diagram. The lesson compresses to a sentence worth carrying into every deployment conversation: a model without a monitoring plan is not an asset; it is a liability with good first-quarter manners.

8.14.3 Build Versus Buy

The third vignette is build versus buy, composited from diligence processes that recur wherever marketing technology is purchased, because most working analysts will meet this chapter's machinery not as builders but as evaluators of other people's claims. The modern marketing stack sells predictions everywhere: customer-data platforms ship churn and spend scores, CRM suites ship propensity models, ad platforms ship value-based bidding — and the buy side is often right, because a vendor amortizing pipeline engineering across hundreds of clients can genuinely out-build a two-person analytics team. What separates a sound purchase from an expensive slide is the diligence, and the diligence is this chapter run as a questionnaire: the frame (what unit, target, and horizon — a "churn score" with an undisclosed churn definition is not yet a product claim, per Section 8.3's label discipline); the evaluation (out of sample on whose data, and how was complexity chosen); the baseline (better than what — scores that barely beat trailing spend are repackaging your own spreadsheet); the leakage audit (what are the features, and can impossible values be range-checked); and the maintenance contract (who monitors, who retrains, what triggers).

The artifact that converts that questionnaire from a conversation into evidence is an acceptance test written into the pilot: the vendor scores a historical snapshot of the client's own customers, chosen by the client, with the outcome window already complete but withheld from the vendor. The client then computes the metrics — capture at the client's own capacity, error in the client's own dollars — against the client's own incumbent rule, using code the client wrote. Performance on someone else's base is drift waiting to be discovered; performance on a vendor-selected sample is a sales artifact; performance on a client-held holdout is the only version that means what the contract will claim. Teams that run this report an unexpected benefit: the vendors worth buying agree to it easily, because the honest ones did the work and would rather demonstrate it than argue about it. The meeting where a vendor cannot name their own snapshot date is a license fee saved.

8.14.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into one expectation: the modeling is the minority of the work. The frame conversations, the label negotiations, the leakage audits, the baseline politics — the spreadsheet's defenders are stakeholders, not obstacles — the adoption design, the monitoring calendar, the vendor diligence: that is the job's actual shape, and the fitted model sits inside it as one well-understood component. The industry's software has made fitting nearly free. What it has not made free, and what this chapter trained, is knowing whether the number the software prints deserves to be believed — and the analysts who advance are the ones whose belief, granted or withheld, turns out to be worth something.

8.15 Ethics, Acting on Predictions About People

The Business Analytics in Practice section described organizations wiring predictions into decisions; this section examines the wiring as an ethical act, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), and causal language (Section 7.15) — to what this chapter adds: decisions made about individual people, automatically and at scale, on the strength of statements about people who resemble them. The chapter's ethics is not a caution about misuse at the margins. Acting on predictions is the technique's entire purpose, and the act carries obligations that the arithmetic, however honest, does not discharge on its own.

Begin with what a score is, because the ethics follows from the epistemology. The model's 585 dollars for customer C004471 is, as Section 8.13 established, a conditional average over historically similar customers, carrying

a typical error the test set measured in tens of dollars. Statistically, that is a fine instrument for building a list. Ethically, the shift happens at the moment of action: the Backstage program does not gift a conditional average — it gifts, or declines to gift, a person, and the person receives the treatment their resemblance earned, not the treatment their actual future would have justified. At the individual grain, some of the model's decisions are simply wrong, in both directions.

It is tempting to say that the test MAE tells you how often. It does not, and the precision matters because this is a chapter about not overstating what a number establishes. MAE is unsigned and it is an average magnitude: it reports how far the model misses on held-out customers, on average, in dollars. It does not report a frequency of wrong decisions, it does not say which side of the cutoff any customer fell on, and it does not distinguish a twenty-dollar miss that changed nothing from a two-hundred-dollar miss that displaced someone from the list. What the test MAE is, exactly, is the honest numerical size of the model's individual-level ignorance. That is enough to dissolve the comfortable idea that a validated model has earned the right to be unexamined — and it is worth being equally precise about what validation bought. Validation did not certify the average in any general sense. It estimated average performance on a stated holdout population, at a stated snapshot, under a stated use, with sampling uncertainty attached and no claim about transport to a later period. The decisions, meanwhile, land one at a time.

Three obligations follow, each with a StyleCraft edge. The first is proportionality between the model's demonstrated reliability and the stakes of the automated act. A capture improvement of a few points honestly justifies routing lookbooks; the same model, repurposed to decide which customers' service complaints get escalated or whose returns get flagged for review, is the same arithmetic carrying stakes it was never validated for — and the repurposing conversation, which will arrive because scores are convenient and licenses are sunk costs, is the analyst's to have. Function creep, not initial deployment, is where scoring programs most often cross their ethical lines.

The second is transparency of the automated targeting itself, in two registers. Internally, the deliverable's frame and monitoring plan are the transparency instrument: anyone operating the program can learn what the score is, what it is not, and how wrong it typically runs — which is why Section 8.13 made the frame page one. Externally, the customers on neither side of the line were asked whether their transaction histories could audition them for gifts, and while a lookbook is benign, the guide's standing test — would you be comfortable explaining the program's mechanics, aloud, to the customers on both sides of the cut? — is worth running precisely because it is easy to pass here. A program that passes it at twenty dollars a slot and fails it when the same scores set service priorities has located its own boundary, and the analyst who ran the test knows where the boundary is before the repurposing meeting starts.

The third obligation is memory that the scores inherit the past. The model learned from behavior shaped by everything StyleCraft has already done — where stores were opened, who was marketed to, which segments got offers — so the scored list partly re-awards yesterday's attention as tomorrow's gifts; the disparate-impact frame introduced with Barocas and Selbst in Section 1.13 applies whenever resemblance-based decisions recycle historical patterns, and the StyleCraft dataset's own designed trap — the newest stores, whose customers score poorly for no reason except the stores' youth — is waiting in exactly this chapter's machinery for any analyst who scores without asking what the training window could not yet know. The eligibility rule of Section 8.3 is where this bites hardest and least visibly: a population defined by having purchased in an eighteen-month window is a population defined, in part, by who was reachable during it.

> **BRIDGE**
>
> Chapter 9 takes the next step: when predictions become yes/no verdicts, fairness across segments and the feedback loops by which models cause what they predict become the central ethical machinery.

What the discipline asks of the analyst, concretely, is a fourth section on the deliverable's last page — after the frame, the leaderboard, and the monitoring plan: the acting-on-it note. Three sentences suffice for the Backstage program: what the score may be used for (ranking customers for this program's treatments); what it may not be used for without revalidation (any decision with materially higher stakes, any population outside the eligibility rule, and — if the leaderboard selected a candidate that ranks well while calibrating poorly — any use of the score as a dollar

estimate); and who reviews the answer when a use-case question arrives. The note costs a paragraph and does a specific job: it converts the analyst's private judgment into the program's standing policy, which is the difference between ethics that depends on the analyst being in the room and ethics that survives their promotion. The analyst of record signs the scores; the note is what the signature means.

CONCEPT

The Average Is Estimated; the Act Lands on a Person

Every predictive deliverable in this guide carries the distinction in writing: what the evaluation estimated (ordering quality and typical error, on a stated population, at a stated snapshot, for a stated use) and what it did not (the correctness of any individual decision, the model's fitness for stakes and populations it was never graded on, or its performance at a later date after the world has moved). Before wiring a score into an automated act, name the act's stakes, check them against the evaluation, and write the permitted-use note.

The test MAE is not a disclaimer to bury and not a count of mistakes; it is the honest size of the model's individual-level ignorance, and the program design — reversible treatments, human review where stakes rise, revalidation before repurposing — is where that ignorance is either respected or laundered.

Source: Course concept developed for this guide, informed by Barocas and Selbst (2016) and Provost and Fawcett (2013).

8.16 Chapter Summary

This chapter crossed the line Chapter 7 drew and the guide had been approaching since Part II began: from models whose coefficients are the deliverable to models whose outputs are, and from grading analyses on their honesty about the past to grading them on their accuracy about a future they have not seen. The main point is a change of posture. Explanatory work earns trust through audited structure — signs, magnitudes, units, diagnostics. Predictive work earns trust structurally: a frame declared before fitting, a wall in time between what the model may know and what it must predict, an opponent named before the race, complexity chosen inside the training data, and a grade computed once, on sealed customers, in the decision's own currency. Every impressive number produced outside that structure is flattery of one kind or another, and the chapter's standing red flag inverted the beginner's instinct in a sentence: in prediction, the result that looks too good almost never is.

The machinery arrived in three movements. The framing movement built the predictive frame — unit, target, features, horizon, and, in this chapter's version, the metrics and baselines declared alongside them — and the label as a manufactured measurement: a snapshot date, a feature window and an outcome window with both ends enforced, an eligibility rule evaluable at the snapshot, and the insistence that a zero is a label rather than a gap. The modeling population was distinguished from the deployment population, and the customers the rule excludes were named as an activation problem rather than absorbed silently. Leakage was named as the canonical error and cataloged by route — temporal, target-derived, outcome-window contamination, split contamination, population — with the audit, not the software, as its only reliable detector, and with the split-contamination rule stated precisely enough to be usable: what contaminates a split is a step that estimates something from data, not any step performed before it.

The evaluation movement built the honest exam. Train/test splits defeat in-sample flattery, and the chapter was careful about what they buy: a random customer holdout estimates cross-sectional generalization at the historical snapshot and does not measure time-forward transport to a later scoring date. MAE and RMSE were read in dollars and separated from the decision metric, because the Backstage program acts through a fixed ranked cutoff and a model can improve its typical miss while worsening the ordering at the line — so top-decile capture leads the frame and MAE leads the calibration reporting. Baselines were operationalized from Section 2.7's interpretive discipline into working opponents, with the incumbent's sorted spreadsheet dignified as the baseline with teeth and the

direction of the error metrics stated plainly, since a model far below the baselines on MAE is the suspicious one and a model far above them is the broken one. Overfitting and underfitting were read jointly from the training-versus-validation signature, with the guarantee attached to training RMSE rather than to training MAE, and with the limit on honest performance described as a documented picture of the outcome's volatility rather than a computed noise floor the chapter had not earned. Cross-validation stopped being a concept and became the instrument: one fold generator, one leaderboard, every candidate and both baselines scored by the same call, the test set sealed throughout.

The deployment movement carried the score into the world: rank and cut under a fixed budget with a declared tiebreaker, the capture head-to-head against the incumbent list, expected-value framing at concept level with the word incremental flagged as unearned, and the deployed model's real life stated in three terms rather than one — data drift in the inputs, concept drift in the relationship, performance decay as the observable consequence — each with its own monitoring implication and its own trigger. The AI section named the assistant-era failures — the convenient-table leak, the celebrated impossible fit, the missing opponent, the metric that does not match the act — and installed the five-point audit that the labs then ran three ways: on the honest pipeline, on a deliberately manufactured leak whose smoking gun was a single impossible range check, and on an assistant's unframed enthusiasm. Sixteen numbered cells carried the argument into executable form: the miniature baselines with their one-slot capture, a snapshot-safe feature builder that takes its window as an argument, both windows enforced at both ends, the eligible population reconciled against the customer file, the labeled table audited before anything was fitted, the split frozen by customer identity, five candidates registered with every learned step inside a pipeline and a selection rule written before the race, the cross-validated leaderboard with its paired fold margins, the flexibility dial turned without touching the test set, the rule executed unedited to name a finalist, that finalist opened against the sealed customers once, the leak manufactured on the same customers and caught by arithmetic, and the two ranked lists compared deterministically. The interpretation, practice, and ethics sections carried the results into the meeting room, where the vendor's 96 percent met the five questions and a vocabulary correction, the score was demoted from verdict to ordering, the capture gap was distinguished from a treatment effect, and the act of gifting eight hundred people on the strength of their resemblance to other people acquired its permitted-use note.

Looking ahead, the chapter's target was a dollar amount, and dollars were a mercy: a numeric miss has a size, and MAE could price it. Most of marketing's predictive questions refuse the mercy. Will this customer churn; will she respond; will he click, convert, return — yes or no, where a prediction is not close or far but right or wrong, and where being wrong has two faces with two different prices. The next chapter takes up classification, propensity, and churn: the logistic machinery that turns features into probabilities, the confusion matrix that gives the two error types their business names, the lift and gains machinery this chapter borrowed one exhibit from, and the threshold decision — set not by statistics but by the asymmetric costs Section 2.7 planted and this chapter previewed — where the analyst's model finally negotiates directly with the CFO's economics.

8.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: declare the frame before touching data, define labels in time with both ends of both windows enforced, trace every feature for leakage, choose complexity inside the training data, grade models only out of sample and only against named baselines, read the training-versus-validation signature, translate scores into decisions with their economics attached, and treat too-good results as audits waiting to be run. They are organized into two groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Assignment #3, Drivers and Predictive Modeling, draws on the homework sets of Chapter 7 and this chapter. In-class activities are prepared for discussion rather than submitted. Each exercise also carries its assignment label — required practice, homework submission, or in-class discussion — so that instructors can assign selectively.

8.17.1 Core Chapter Practice

Exercise 8.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

53. State what changes when a model's outputs, rather than its coefficients, become the product, and name one practice that is acceptable in explanatory work and a failure in predictive work.
54. Define the four commitments of the predictive frame, and explain why the decision metric and the baselines must be declared before fitting rather than after.
55. Explain why “future spend” is a manufactured label rather than an existing column, and state why each of the four declared dates is load-bearing rather than decorative.
56. State this chapter's eligibility rule, explain why both of its conditions can be evaluated at the snapshot, and name the population it excludes and what should happen to that population.
57. Define leakage and explain why it cannot be detected by fitting statistics or by a train/test split, only by audit. What is its most reliable symptom?
58. A colleague says that any transformation applied before the split contaminates it. Give one transformation for which this is true and one for which it is false, and state the test that distinguishes them.
59. An analyst reports that their model's training MAE is 31 dollars and its five-fold cross-validated MAE is 78 dollars. Diagnose, using Table 8.4's vocabulary, and prescribe.
60. Why does this guide compute the mean baseline's prediction from the training labels only, and which rule from Section 8.5 is that the smallest example of?
61. RMSE substantially exceeds MAE on the same predictions. What does the gap announce, and which StyleCraft customers — by Chapter 5's findings — are almost certainly responsible?
62. Explain why capture and MAE can disagree about which candidate is better, and state what the deliverable should do when they do.
63. State the three clauses of the model-comparison discipline, and for each, describe the loophole it closes.
64. Distinguish data drift, concept drift, and performance decay, and state which of the three can be detected without waiting for outcomes to mature.

Exercise 8.2 Draw the Frame (Required Practice)

For each brief below, write a complete predictive frame — unit of prediction, target with a fully defined label (snapshot, feature window with both ends, outcome window with both ends, filters, eligibility), horizon, decision metric, calibration metrics, and the strongest honest baseline — and then name the single most likely leakage route from Table 8.2 and how your frame forecloses it.

65. The CRM manager wants to time win-back email for customers likely to lapse: predict each customer's days until next purchase.
66. Merchandising wants to know, at the end of each customer's first month, what they will spend over their first year, to set onboarding investment.
67. The email team wants to predict, per customer, next quarter's email-driven revenue — and someone proposes using each customer's full-history open rate as a feature. Address the proposal explicitly.
68. The retention team wants to score every customer monthly, including customers acquired last week. State what your frame can and cannot do for the newest customers, using the representation argument of Section 8.3 rather than a claim about technical impossibility.

Exercise 8.3 Baseline Arithmetic on the Miniature (Required Practice)

Using only the ten-customer miniature of Lab 8.1 Part A (no code): verify the mean baseline's MAE of 53.84 by computing all ten absolute errors and summing; verify the last-value baseline's MAE of 84.80 the same way; identify the four customers responsible for the incumbent rule's defeat and state, for each, the behavioral profile from Chapters 5 and 6 that explains the miss; explain why the RMSE gap between the two baselines (161.80 versus 87.42) is proportionally wider than the MAE gap, and what that says about how the incumbent fails. Then do the decision arithmetic: with two slots rather than one, list which two customers each rule selects and what share of the

ten customers' outcome-window spend each pair captures. Close with one sentence on why the capture comparison is a harsher verdict on the incumbent than the MAE comparison, and one on what a model must do to beat both baselines at once.

Exercise 8.4 Spot the Leak (Required Practice)

Each pipeline below contains at least one leakage route from Table 8.2, or none. Name the route (or certify the pipeline clean), cite the table row, and state the repair.

69. To predict spend_next6m from a December 31 snapshot, an analyst includes loyalty_tier as of the June 30 database extract, reasoning that tier changes slowly.
70. A churn model's training population is “customers with at least three orders in the 24-month window,” with churn labeled in the final six months.
71. An analyst standardizes all features on the full labeled table, then splits 80/20, fits, and reports test MAE against a mean baseline computed from training labels.
72. An analyst expands two predictors into third-degree polynomial terms on the full labeled table, then splits 80/20 and fits. Be careful with this one, and justify your verdict with the test from Section 8.4 rather than with the position of the split.
73. To predict first-year spend at the end of month one, the feature list includes number of orders in the first 30 days and average order value over the customer's lifetime.
74. A team refits Chapter 6's segmentation on all labeled customers, adds the resulting segment indicators as features, then splits and cross-validates the model. Name what leaked, and state the two structural repairs Section 8.9 offers.
75. An assistant's pipeline for the Backstage frame reports test R^2 of 0.95, and its feature importance list is led by a column named monetary.

Exercise 8.5 The Full Predictive Deliverable (Homework Submission)

Complete Labs 8.1 and 8.2 on the certified files and assemble the scoring deliverable per Section 8.13: the one-page frame with its seven declarations and four dates; the reconciled population counts, naming the excluded groups; the cross-validated leaderboard with both baselines and every candidate's margin, computed on the training customers only; the single test-set table with capture leading and MAE and RMSE beside it; the capture head-to-head against the incumbent list with the dollar gap and the count of names that change; the expected-value sketch with the incremental caveat stated and the holdout named; the vendor-verdict memo built on Table 8.5's five questions; the monitoring plan with its inputs, its outcome benchmark, and its retraining trigger; and the permitted-use note per Section 8.15. Submissions are graded on the frame's completeness, the order of operations evidenced in the notebook, and the audit evidence as heavily as on the code.

Exercise 8.6 AI Pipeline Audit (Homework Submission)

Give an AI assistant the raw StyleCraft tables and this deliberately loose prompt: “Build me the best possible model to predict customer spend — maximize accuracy.” Then audit the response with Table 8.5's five points, in order and in writing: reconstruct the frame the assistant implicitly chose and compare it to Table 8.1; trace every feature to its window, running the recency range check if applicable; check the split and preprocessing hygiene, including whether any complexity or tuning decision was made by consulting a test set; determine what baseline, if any, the pipeline was compared against, and compute the missing ones yourself on the same customers; and grade the reported performance for plausibility against the volatility figures you recorded in Code 8.6, stating whether the assistant's own narration celebrated or questioned it. Separately, list every performance claim in the reply and mark each as earned under exam conditions or not, flagging any use of classification vocabulary for a numeric target. Document the full exchange per the AI-use documentation template in Appendix D, and conclude with two sentences on which audit point caught the most serious problem.

8.17.2 In-Class Activities

Exercise 8.7 The Vendor Meeting (In-Class Discussion)

Your team will role-play the diligence meeting for the vendor's predictive-spend module. Prepare: the five questions of Table 8.5 translated into vendor-facing language; a one-sentence correction of the “96 percent accuracy” claim that a non-technical buyer could repeat, distinguishing accuracy from R^2 and noting what an out-of-sample R^2 can do that a percentage cannot; for each question, the answer an honest vendor would give and the answer that should end the meeting; the acceptance test you would write into the pilot, per Section 8.14's third vignette, naming who holds the outcome window and who computes the metrics; a position on what evidence would justify the license fee over the in-house model, expressed as a capture margin in dollars; and one question about the maintenance contract that Section 8.14's second vignette proves is not optional. Half the class prepares the vendor's side; the honest version of that brief is harder, and more instructive, than the buyer's.

Exercise 8.8 Find the Flaw in the Predictive Pipeline (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

76. A team celebrates a spend model whose test MAE beats the mean baseline by 40 percent. No other baseline appears in the deck, and the CRM team's trailing-spend rule is not mentioned.
77. An analyst fits polynomial models of degree 1 through 9, plots training and test error for each, picks the degree where test error is lowest, and reports that degree's test error as the honest out-of-sample estimate.
78. An analyst tunes a model through thirty rounds of adjust-and-check against the test set, then reports the final test MAE as the honest out-of-sample estimate.
79. A model's test MAE of 51 dollars is reported as “the model is wrong by 51 dollars about every customer, so it cannot be trusted for individual decisions” — and the program is cancelled.
80. A deck reports that the model's list captures 6 percentage points more spend than the incumbent's and concludes that the program will generate that much additional revenue.
81. A pipeline predicts six-month spend with test R^2 of 0.97; the lead analyst schedules a celebration and a production launch for the same week.
82. A deployed value model performs beautifully for three quarters; monitoring consists of confirming the scores arrive on schedule. A repricing launches in month ten.
83. A comparison table shows Model A winning on MAE and Model B winning on RMSE; the report recommends Model B because “it wins on the more sophisticated metric.” The frame declared capture first and MAE second.
84. Two analysts run the same notebook on the same data and produce eight-hundred-name lists that differ by fourteen customers. Nothing in either notebook errored.

8.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Baseline model. A deliberately simple prediction rule — mean, majority class, or last value — evaluated on the same folds, the same test set, and the same metrics as a candidate model, whose margin of defeat is the measure of the model's demonstrated value (adapted from Provost & Fawcett, 2013).

Bias–variance tradeoff. The intuition that prediction error decomposes into systematic error from oversimplification (bias) and instability from fitting sample-specific noise (variance), with model flexibility trading one for the other and honest performance minimized between the extremes (adapted from James et al., 2021; Domingos, 2012).

Concept drift. Change over time in the relationship between a model's inputs and its target, so that the learned mapping no longer describes how the outcome is generated; invisible in the input distributions and detectable only as outcomes mature (adapted from Gama et al., 2014).

Cross-validation. Estimation of out-of-sample performance from training data alone, by k-fold rotation of a held-out fold and averaging of the k held-out scores; the standard instrument for development-time decisions, including model complexity, while the test set stays sealed (adapted from James et al., 2021; scikit-learn developers, 2026b).

Data drift. Post-deployment change in the distribution or quality of a model's input features, so that scored cases increasingly differ from training cases; detectable by watching the inputs alone. Covariate shift is its special case, in which the predictor distribution changes while the conditional relationship to the outcome remains stable (adapted from Gama et al., 2014).

Expected value framing. The translation of predictions into actions by weighing each action's possible outcomes by their probabilities and subtracting its costs, so that score thresholds become economic lines rather than arbitrary ranks (adapted from Provost & Fawcett, 2013).

Feature window. The declared span of time, bounded at both ends and closing at the snapshot date, from which a predictive model's features may be computed; the past side of the label's wall in time (adapted from Kuhn & Johnson, 2013).

Horizon. The forward distance between the snapshot date and the end of the outcome window: how far into the future the model's claim extends, constraining validation timelines and retraining cadence (adapted from Provost & Fawcett, 2013).

In-sample flattery. This guide's name for the systematic optimism of performance measured on the data a model was fitted to, which grows with model flexibility and cannot be estimated from the training data itself (adapted from James et al., 2021).

Label. The constructed, recorded value of the target for a historical case, manufactured by choosing a snapshot date, a feature window, an outcome window, and eligibility rules; the answers a predictive model trains on (adapted from Provost & Fawcett, 2013; Kuhn & Johnson, 2013).

Leakage. The presence in training features or evaluation procedure of information unavailable at real prediction time — from the future, from the target's construction, or from the evaluation data — inflating measured performance that collapses on deployment (adapted from Kaufman et al., 2012).

MAE (mean absolute error). The average absolute difference between actual and predicted values, in the target's native units, weighting all misses equally; an unsigned average magnitude rather than a count of wrong decisions, and this guide's lead calibration metric (adapted from Willmott & Matsuura, 2005).

Outcome window. The declared span of time after the snapshot date, bounded at both ends, over which the label is measured; the future side of the wall, which no feature may touch (adapted from Kuhn & Johnson, 2013).

Overfitting. A model's capture of training-data-specific noise that does not generalize, diagnosed by training performance substantially exceeding held-out performance and worsening with added flexibility (adapted from James et al., 2021; Domingos, 2012).

Performance decay. The erosion of a deployed model's predictive quality over time as data drift, concept drift, or both accumulate, observed by monitoring realized error as fresh outcomes mature (adapted from Gama et al., 2014; Sculley et al., 2015).

Pipeline. An object chaining every learned step of a model — scaling, imputation, clustering, the estimator itself — so that all of them are refitted together on each training fold; the structural defense against split contamination (adapted from scikit-learn developers, 2026a).

Predictive frame. The four declared commitments of a predictive model — unit of prediction, target, features, and horizon — written before fitting, together with the decision metric, the calibration metrics, and the baselines, as the predictive extension of the analytic specification (adapted from Provost & Fawcett, 2013; Kuhn & Johnson, 2013).

Predictive modeling. The construction of models whose purpose is accurate output for individual unseen cases, evaluated out of sample, as distinct from explanatory modeling graded on the honesty of its coefficients (adapted from Shmueli, 2010; Provost & Fawcett, 2013).

RMSE (root mean squared error). The square root of the average squared difference between actual and predicted values, in native units but disproportionately sensitive to large misses; its excess over MAE announces error concentration (adapted from Willmott & Matsuura, 2005).

Snapshot date. The declared moment that splits time in a label's construction: features may use only information available on or before it; the label is measured only after it (adapted from Kuhn & Johnson, 2013).

Split contamination. The leakage route in which a step that estimates something from data is fitted before the split, letting held-out cases influence the values the model trains on; distinguished from transformations that estimate nothing and are therefore unaffected by the order of operations (adapted from Kaufman et al., 2012; scikit-learn developers, 2026a).

Test set (holdout). The randomly held-out portion of labeled cases, sealed during development and consulted once for the selected model, whose performance estimates cross-sectional out-of-sample error at the historical snapshot (adapted from James et al., 2021).

Capture (at a selection share). The share of a population's actual outcome-window spend held by the customers a model ranks inside the program's capacity — the capacity divided by the population being ranked. It is the decision metric of a fixed-capacity ranked targeting program, and the metric on which a model's list is compared with an incumbent's; where the share is a tenth it is the familiar top-decile case. A constant score carries no ranking content, so its capture is defined only once ties are broken on a declared rule, and a random tiebreak makes that figure the selection share itself (adapted from Provost & Fawcett, 2013).

Training set. The portion of labeled cases from which models, every learned preprocessing step, and every development decision including model complexity are derived (adapted from James et al., 2021).

Underfitting. A model's failure to capture structure that does generalize, diagnosed by poor performance on training and held-out data alike (adapted from James et al., 2021).

8.19 Further Readings

Students who want additional background may begin with the following readings. The leakage and framing sources are listed first, methods second, software and deployment last.

- Kaufman et al. (2012) for the definitive treatment of leakage — its formulation, its taxonomy, and the case studies from data-mining competitions where it repeatedly crowned false winners; the extended companion to Section 8.4.
- Provost and Fawcett (2013), especially the chapters on model evaluation and the expected-value framework, for the business-first treatment of everything from baselines to score-to-decision translation; the closest published relative of this chapter's approach.
- Domingos (2012) for twelve compact pages on what practitioners most often get wrong — overfitting, the futility of accuracy without generalization, and why more sophisticated is not a direction; the fastest inoculation in the field's literature.
- James et al. (2021), Chapters 2 and 5, for the standard accessible development of the bias–variance tradeoff, validation, and cross-validation, one level of formality above this chapter — and, for the implementation of exactly those ideas, the two pages of official scikit-learn documentation these labs are built on (scikit-learn developers, 2026a, 2026b): the common-pitfalls page on preprocessing leakage and pipelines, and the cross-validation page on using folds for development while a test set is reserved for the final grade.
- Willmott and Matsuura (2005) for the short, pointed argument on MAE versus RMSE and what each actually measures; useful ammunition for the metric-choice conversation.
- Sculley et al. (2015) for the famous statement that the model is the smallest box in the system diagram — technical debt, entanglement, and why deployment is where predictive work is won and lost; the companion to Section 8.10.

8.20 References

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM, 55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*(4), Article 44. https://doi.org/10.1145/2523813

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, methodology, and avoidance. *ACM Transactions on Knowledge Discovery from Data, 6*(4), Article 15. https://doi.org/10.1145/2382577.2382579

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer. https://doi.org/10.1007/978-1-4614-6849-3

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

scikit-learn developers. (2026a). *Common pitfalls and recommended practices* [Software documentation]. https://scikit-learn.org/stable/common_pitfalls.html

scikit-learn developers. (2026b). *Cross-validation: Evaluating estimator performance* [Software documentation]. https://scikit-learn.org/stable/modules/cross_validation.html

scikit-learn developers. (2026c). *PolynomialFeatures* [Software documentation]. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

scikit-learn developers. (2026d). *r2_score* [Software documentation]. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems 28* (pp. 2503–2511). Curran Associates.

Shmueli, G. (2010). To explain or to predict? *Statistical Science, 25*(3), 289–310. https://doi.org/10.1214/10-STS330

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research, 30*(1), 79–82. https://doi.org/10.3354/cr030079

Chapter 09

Classification, Propensity, and Churn Models

From How Much to Whether: Probabilities, Verdicts, and the Price of Being Wrong in Both Directions

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops classification — the prediction of yes/no outcomes — as the completion of the machinery Chapter 8 built for numeric targets, with churn as its application. It constructs the churn label as a measurement decision, builds logistic regression from the log-odds equation, and adds classification trees and k-nearest neighbors as structurally different scorers compared on a training-only cross-validated leaderboard. It then builds the evaluation vocabulary: the confusion matrix in campaign terms, accuracy's failure under imbalance, precision and recall as trade-off instruments, ROC and AUC as threshold-free measures of ranking skill, and calibration as the separate property an economic threshold consumes. The destination is the decision threshold derived from asymmetric error costs, with propensity scoring, deciles, lift, and gains translating scores into budget-sized lists. The labs verify every figure by hand, keep the test set sealed until a finalist is chosen, and close with a group-level fairness screen.

KEYWORDS

classification; churn; logistic regression; confusion matrix; precision; recall; ROC/AUC; decision threshold; propensity; lift

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Classification, propensity, and churn models. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 9, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, NumPy, Matplotlib, and scikit-learn are sponsored or affiliated projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. statsmodels is a community-developed project distributed under the modified BSD license. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter are in the Applied Business Analytics companion repository on GitHub.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors,

researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

- Identify marketing's binary prediction questions, distinguish classification from class-probability estimation, and explain at intuition level why linear regression is the wrong instrument for a binary target.
- Define a churn label completely for a non-contractual business — activity window, activity definition, eligibility, and the snapshot structure of Section 8.3 — and explain why the label is a measurement decision with program consequences, per the four-element discipline of Section 3.5.
- Explain the logistic function and log-odds at intuition level, fit and read a logistic regression, and translate its coefficients into plain-language statements about odds and probabilities without overclaiming causation.
- Describe how classification trees and k-nearest neighbors score customers, state each method's characteristic strengths and failure modes, and explain why k-NN requires the feature scaling of Section 6.7.
- Construct a confusion matrix from a scored list and a threshold, name its four cells in business vocabulary, and verify its arithmetic — counts, accuracy, precision, recall — by hand.
- Explain accuracy's failure under class imbalance, compute the majority-class baseline of Section 8.7 as the working floor, and state the imbalance-handling options at concept level.
- Compute and interpret precision, recall, and F1 in campaign terms, and choose which to lead with based on the decision's cost structure.
- Read ROC curves and AUC at working level as measures of ranking skill, state the full range AUC occupies and what it does and does not certify, and treat implausibly high AUC as the leakage signal of Section 8.4.
- Distinguish a model's ranking skill from its probability calibration, evaluate calibration out of fold on training data, and state the condition calibration imposes on any economically derived threshold.
- Select a model family and its settings using stratified cross-validation inside the training customers, per Section 8.9, and open the sealed test set once for the finalist.
- Derive a decision threshold from stated false-positive and false-negative costs using the expected-value framing of Section 8.10, explain why the software's default of 0.5 is an inheritance rather than a decision, and reconcile an economic threshold with a budget constraint.
- Build ranked-targeting exhibits — decile tables, lift charts, and gains charts — and translate them into a treatment recommendation and a budget argument.
- Verify an AI-built or vendor-supplied classifier with the five-point audit of Section 8.11 plus the classification supplement — base rate and majority baseline, hand-checked confusion matrix at the economic threshold, threshold re-derived from stated costs, calibration evidence — documenting the audit per Appendix D.
- Examine a classifier's error rates across customer groups, explain how classification models can cause the outcomes they predict, and state the fairness and feedback-loop obligations that attach to acting on classifications at scale.

Chapter 8 ended with a mercy it named on its way out. The Backstage deliverable predicted a dollar amount, and a dollar amount can be missed by a little or by a lot: the mean absolute error priced the typical miss, the leaderboard priced the model's margin over the sorted spreadsheet, and the whole apparatus of honest evaluation ran on the fact that a numeric error has a size. Most of marketing's predictive questions refuse the mercy. Will this customer churn; will she respond to the offer; will he click, convert, come back — yes or no, where the decision the model feeds is not close or far but right or wrong, and where being wrong has two faces with two different prices. This chapter builds the machinery for those questions: the logistic model that turns features into probabilities, the trees and neighbors that offer second opinions, the confusion matrix that gives the two kinds of error their business names, the curves that grade a model's ordering skill, the calibration check that grades its probabilities, and — the chapter's destination, and the payoff of a thread planted seven chapters ago — the decision threshold, chosen not by

statistical convention but by the asymmetric costs of the two mistakes. Every discipline Chapter 8 installed rides along unchanged: the frame is still declared before fitting, the label is still a manufactured measurement, leakage is still the canonical error, model selection still happens inside the training data, and the test set is still touched once. What is new is that the number the model produces is a probability, the number the meeting wants is a verdict, and the space between them is where this chapter's analyst earns the seat at the table.

CONCEPT

What This Chapter Is Really About

Chapter 8's machinery graded predictions by how far they missed. This chapter's *verdicts* cannot miss by a distance — a customer predicted to churn either does or does not — and that small change of geometry rearranges everything downstream. One number can no longer summarize performance, because a binary verdict fails in two distinguishable ways, and the two failures bill different departments: the offer wasted on a customer who was staying anyway is a marketing cost, and the customer lost because no one intervened is a revenue cost, and nothing in the mathematics makes them equal. The chapter's methods, however, do not produce verdicts. They produce probabilities, and a probability forecast can still be more or less wrong — predicting 0.90 for a customer who stays is a larger probabilistic error than predicting 0.55, and 0.51 and 0.99 yield the same verdict at a 0.5 cut while carrying very different confidence. Class verdicts therefore create two error types; probability forecasts additionally require evaluation of their ranking and their calibration, and this chapter grades all three.

The chapter's real subject is the sequence of separations this forces on the analyst: the score separated from the verdict, because the model produces a probability and someone must choose where to cut it; the model's ranking separated from its calibration, because a well-ordered list of scores can still be systematically too high or too low; the model's quality separated from the threshold's wisdom, because a well-ranked list can be cut in a foolish place; and the statistical output separated from the economic decision, because the threshold that converts scores into treatments is not a modeling parameter at all — it is a business policy, derivable from the costs of the two errors, and the analyst who cannot re-derive it from stated costs has delegated the most consequential number in the pipeline to a software default. Chapter 2 planted the idea that the cost of being wrong is asymmetric. This chapter is where that idea stops being a lens and becomes arithmetic.

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013).

9.1 Marketing Decision Context: The Save Offer and the Default Threshold

Chapter 8's deliverable shipped, and the Backstage Preview list was only the top of the fall retention program. Its other end concerns the customers StyleCraft is losing. Chapter 5's certified findings put a number on the worry that opened this guide: repeat-purchase rates are weakest exactly where the expansion bet is placed, and a large fraction of customers who buy once never buy again. The program's answer is a save track — internally, the Comeback Edit: a personalized offer combining a $15-off-$75 voucher, free shipping, and a curated selection drawn from the customer's purchase history. Finance has costed the treatment at roughly twelve dollars per treated customer once redemption rates and margin give-up are blended. The question is the list. Who gets the Comeback Edit — and, harder, who decides who gets it, on what evidence, cut at what line? The save track goes live with the November journeys; the targeting rule must be locked in twenty-eight days.

Three answers are already circulating, and each conceals a failure this chapter exists to expose. The first is the incumbent rule, defended by the same CRM operations lead whose sorted spreadsheet fought honorably in Chapter 8: "Anyone with no purchase in ninety days is lapsing. Send the offer to the ninety-day-lapsed list. We have run

win-back this way for two years." The second lives in the marketing platform StyleCraft already licenses, whose Churn Risk panel sits one click from activation behind the claim that ends meetings: "Our churn model is 89 percent accurate." Activate the flag, treat the flagged, done. The third comes from the growth lead, and it has the charm of arithmetic honesty: "Email is nearly free. Why are we rationing a retention offer at all? Send the Comeback Edit to everyone who has ever bought. You cannot mis-target a blanket." The CFO, attending because the offer is margin and margin is the CFO's, has a preemptive answer: the voucher is not an email. Every redemption is real money, a blanket send prices out at twelve dollars times the whole file, and the finance model says most of that spend would land on customers who were never leaving.

Each answer has a real virtue, and stating the flaws is this chapter's tour. The ninety-day rule is cheap, transparent, and — the labs will show — genuinely informative, because long recency is the strongest churn signal in StyleCraft's data by design. It is also a one-variable model wearing a policy costume, and its one variable is precisely the one Chapter 5's lumpiness findings booby-trapped: the Suburban Occasion cohort routinely goes ninety quiet days between legitimate seasonal baskets, so the rule sprays vouchers across customers mid-cadence while missing the discount-dependent buyer whose recent small purchase disguises an exhausted relationship. The platform's 89 percent invites the question Chapter 8 trained: accurate against what, measured how, at what threshold? This chapter adds a sharper one. If roughly seventy percent of eligible customers do not churn, then seventy percent accuracy is available for free by predicting that nobody churns, so 89 percent is a nineteen-point margin over the do-nothing rule — a margin that could be genuinely valuable or could be produced by the wrong mixture of errors, and the number cannot be interpreted at all without the base rate, the confusion matrix, the threshold, and the cost structure beside it. That is the chapter's real complaint about accuracy: not that 89 percent necessarily fails, but that accuracy alone cannot establish whether the model helps the retention program. And the platform's flag arrives as verdicts, not scores, which means someone else's threshold — almost certainly the software default of 0.5 — has already made StyleCraft's most consequential retention decision without attending a meeting. The blanket proposal fails on the CFO's arithmetic, but its deeper failure returns in Section 9.14: treating everyone is not a neutral act with wasted cost, because a retention offer reaches customers it can annoy, train to wait for discounts, or — the industry's most uncomfortable finding — remind to leave.

So the VP of Marketing has commissioned the analyst — still you — with a deliverable in five parts. A churn definition in writing, because "churn" is not a column StyleCraft possesses but a measurement the analyst must construct and defend. A scored model, selected inside the training customers and graded once out of sample against stated baselines including the ninety-day rule — but graded with a new vocabulary, because a yes/no decision fails in two directions and one accuracy number hides the difference. A threshold derived from the program's own economics, shown as arithmetic the CFO can re-run on a napkin and accompanied by the calibration evidence that entitles the arithmetic to be believed. A ranked list with its lift and gains exhibits, because the budget conversation will not be "is the model good" but "how deep into the file do we fund." And a fairness appendix the VP did not ask for but will need, because the scored file includes the customers of the newest stores, and Section 9.15 will show that a naive churn model has already made up its mind about them for reasons that have nothing to do with their loyalty. The chapter builds those five parts in order: the target and the label, the scorers, the evaluation vocabulary, the threshold and the ranked-targeting exhibits, then the AI audit, the labs, and the three sections that rehearse the meeting.

9.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The ninety-day rule is both a candidate answer and, in Chapter 8's terms, a baseline with teeth. State what the rule would have to be beaten *at* — which metric, on which population — for the model to earn the budget line, and why "accuracy" is not an adequate answer.
- The platform's "89 percent accurate" claim needs several numbers beside it before it means anything. Name at least three, and name the section of this guide that equips you to demand each.
- The Comeback Edit costs about twelve dollars per treated customer. Describe the two ways a targeting rule can waste that money — one for each direction of error — and explain why the two wastes do not cost the same amount. Which earlier section of this guide predicted this asymmetry would become the central question?

The blanket proposal cannot mis-target anyone, by construction. List two costs of treating a customer who was never going to leave, beyond the twelve dollars — and keep the list; Section 9.14's first vignette will add a third cost that surprises most students.

9.2 From How Much to Whether: Binary Targets and Why the Line Fails

Chapter 8 defined the predictive frame — unit, target, features, horizon — and nothing in that definition required the target to be a dollar amount. The frame machinery transfers whole; one sentence of reminder suffices, and the reader who wants the full apparatus has Section 8.2. What changes in this chapter is the target's type, in exactly Section 3.4's sense: spend_next6m was a ratio-level quantity, and the targets of this chapter are binary categoricals — two classes, conventionally coded 1 for the class the business is watching for (the positive class) and 0 for its complement. Marketing manufactures such targets constantly, because marketing actions are mostly binary at the customer grain: send or do not send, and the customer responds or does not; show the ad or do not, and she clicks or does not; the contract month arrives, and he renews or does not. Will she churn, will he respond, will they convert, will it bounce — the yes/no questions are not a special case of marketing prediction; they are its majority.

Two tasks hide inside "predict a yes/no outcome," and separating them now saves confusion for the rest of the chapter. Classification, strictly, assigns each case a predicted class: this customer will churn, that one will not. Class-probability estimation assigns each case a probability of the positive class: this customer's churn probability is 0.62, that one's is 0.08. The second task is the one this guide's methods actually perform and the one marketing decisions actually need, because a probability preserves the ordering and the uncertainty that a verdict throws away — and because, as Section 9.9 will show, the conversion from probability to verdict is a business decision that should not be made inside the model at all. The distinction sounds academic and is the chapter's load-bearing wall: nearly every misuse this chapter catalogs, from the platform's pre-thresholded flag to the assistant's celebrated accuracy, begins by collapsing the two tasks into one.

DEFINITION

Classification and Class-Probability Estimation

Classification is the prediction of a categorical target: assigning each case to one of a fixed set of classes, in this guide's scope two. Class-probability estimation is the prediction, for each case, of the probability of the positive class — a score between 0 and 1 that ranks cases by likelihood and defers the assignment of verdicts to a separately chosen threshold. The methods of this chapter estimate probabilities; classifications are produced from them by thresholding, and the threshold is a decision, not a model property.

Source: Adapted from Provost and Fawcett (2013).

Why not simply reuse Chapter 8's instrument? Linear regression will run on a 0/1 target without complaint — the software fits, the coefficients print, and the output even has a respectable name in older literature (the linear probability model). The failure is not that it refuses to work; it is that it works badly in ways an intuition-level argument makes visible. First, the line does not know the target's boundaries: fit spend-style regression to a 0/1 churn outcome and it will cheerfully predict churn values of 1.3 for the most lapsed customers and −0.2 for the most engaged, numbers that cannot be probabilities and cannot be read as anything else. Second, the line's constant slope is the wrong shape for how binary risk behaves: a straight line says that thirty additional days of recency change churn risk by the same amount whether the customer is at the engaged extreme, in the uncertain middle, or already almost certainly gone — but risk near the floor and ceiling has almost nowhere to move, and the real action lives in the middle. The curve that honest binary risk follows is flat near 0, steep in the middle, and flat again near 1 — an S-shape, and Section 9.4's entire contribution is a principled machine for producing it. Third, the errors of a binary target misbehave in ways that undermine the least-squares machinery of Section 7.6 — a technical point this guide leaves at one sentence, because the first two arguments already carry the verdict: the right instrument for a bounded, S-shaped quantity is a model built for bounded, S-shaped quantities.

One more piece of vocabulary completes the setup, and it is a number rather than a method: the base rate, the share of the positive class in the population — the fraction of StyleCraft's eligible customers who actually churn in the outcome window. The base rate is the single most important number in any classification project and the one most reliably missing from celebrated results. It is the context that makes every other metric legible: an accuracy of 89 percent means one thing when churners are half the file and something entirely different when they are a tenth of it; a model that finds churners at twice the base rate is doing real work, and the same model quoted without the base rate is doing unverifiable work. This guide's rule, enforced in every lab and exercise: no classification result is reported without its base rate in the same exhibit, and Section 9.6 will show what happens to the analyst who forgets — the majority-class baseline of Section 8.7, which sat politely at concept level through the numeric chapter, is about to start winning arguments.

9.3 Defining Churn: The Label as a Measurement Decision

Section 8.3 established that a predictive label is manufactured, not found: a snapshot date splits time into a feature window the model may know and an outcome window the label summarizes, eligibility rules decide who gets a labeled row, and the whole construction is a measurement act carrying Chapter 3's obligations. That machinery transfers to this chapter unchanged — same snapshot of December 31, 2025, same eighteen-month feature window opening July 1, 2024, same January-through-June 2026 outcome window, same two-condition eligibility rule — and none of it is re-derived here; Section 8.3 is the reference. What this section adds is the problem that makes churn the most definitionally treacherous label in marketing: for StyleCraft, churn does not exist until the analyst defines it, and every element of the definition changes who the model learns to find.

The reason is structural, and it divides the industry's retention problems into two families. In a contractual business — wireless plans, streaming subscriptions, gym memberships — churn is an event: the customer cancels, the relationship has a legal off switch, and the data records the date it was flipped. In a non-contractual business like StyleCraft's, there is no switch. No customer ever tells an apparel brand she has left; she simply does not come back, and "has not come back yet" and "is never coming back" look identical in the transaction log for months. Chapter 5 measured this at the descriptive grain — repeat rates and their windows were metric constructions, per the four-element discipline of Section 3.5 — and this chapter inherits the problem at the label grain: churn must be *defined into existence* as a threshold on observed silence, and the definition is a choice among defensible alternatives, not a discovery of a fact.

DEFINITION

Churn and the Churn Label

In a non-contractual business, churn is the sustained cessation of a customer's purchasing relationship, operationalized rather than observed: a customer is labeled as churned when a defined activity — usually a completed purchase — fails to occur within a defined outcome window following the snapshot date. The churn label therefore requires every element of the label construction of Section 8.3 plus a definitional commitment specific to churn: what counts as activity, how long the silence must last to count as departure, and which customers are eligible to be labeled at all. Different defensible definitions produce different labels, different models, and different treated populations.

Source: Adapted from Neslin et al. (2006) and Provost and Fawcett (2013).

The StyleCraft definition, stated in the four-element form of Section 3.5 because a label is a metric with a modeling job: churned equals 1 for an eligible customer with zero completed purchase orders in the outcome window, January 1 through June 30, 2026. Numerator concept: customers with no outcome-window purchase order. Denominator: none — it is a per-customer flag, not a rate. Window: the six-month outcome window, inherited from the Section 8.3 snapshot structure. Filters and eligibility: the Chapter 8 rule in both of its conditions — the

customer's signup_date falls on or before December 31, 2025, **and** the customer completed at least one purchase between July 1, 2024 and December 31, 2025 — so that every labeled customer has a feature-window history to be scored on. Both conditions are load-bearing, and the second is the one drafts drop. Recency, average order value, discount share, and store share are all undefined for a customer with no feature-window purchases, because each divides by a quantity that is zero; a population defined only by signup date would therefore carry rows whose features do not exist. Customers who signed up and never purchased remain a named activation population, counted and reported, outside this churn model entirely.

One implementation detail carries more weight than its length suggests, and the labs enforce it. It is tempting to build the churn flag as the exact complement of Chapter 8's spend label — churned when spend_next6m equals zero — because the two are usually the same customers and the code is one line shorter. They are not guaranteed to be the same customers. Zero net revenue is not the same event as zero purchase activity: an order can net to zero after a discount and a returned line, purchases can be offset by returns and adjustments, and a customer can generate negative net revenue while having unambiguously transacted. The definition above says *activity*, so the label is built from counted purchase orders in the outcome window, and the reconciliation against spend_next6m is reported as a separate diagnostic rather than assumed. Where the two disagree, the disagreement is a finding about the revenue column, not a bug in the label — and the deliverable should not claim the churn flag is the exact complement of the spend label unless the dataset specification guarantees that every completed purchase produces strictly positive net revenue.

Every element of the definition is a lever, and moving any one of them changes the program. Table 9.1 walks the three that matter most. Shorten the outcome window to ninety days and the label sweeps in every Suburban Occasion customer sitting mid-cadence between seasonal baskets — the base rate inflates, the model learns to find slow shoppers rather than departing ones, and the Comeback Edit ships vouchers to customers who were three weeks from a full-price occasionwear order. Lengthen it to a year and the label grows more truthful about permanent departure but the program loses its purpose, because a customer silent for twelve months is far harder to save than one silent for five — the definition trades label purity against intervention timing, and the trade is a marketing judgment, not a statistical one. Redefine activity to include non-purchase engagement — an app open, an email click — and the label stops measuring commercial churn and starts measuring attention churn, a different construct that may or may not be what a voucher can treat. None of these alternatives is wrong. What is wrong is choosing one silently, and the deliverable's first page therefore states the definition, the rejected alternatives, and the reason — the same discipline the metric disputes of Chapter 3 taught, now with a model and a budget downstream of the choice.

Table 9.1

The churn definition's levers and what each one changes

Definitional lever	This guide's choice	What a different choice would change
Outcome window length	Six months (the Section 8.3 outcome window)	Shorter inflates the base rate with mid-cadence occasion shoppers; longer purifies the label but treats customers too late to save
What counts as activity	At least one completed purchase order	Engagement-based definitions measure attention, not commerce; a net-revenue definition mislabels zero-value and fully returned orders as silence
Eligibility	Signed up on or before the snapshot **and** at least one feature-window purchase (Section 8.3)	Dropping the purchase condition admits never-activated signups whose ratio features are undefined, and changes both the base rate and the treated population

One consequence deserves its own paragraph because it will resurface in three later sections. The definition makes the base rate a *constructed* number: StyleCraft's churn rate under this label is not a fact about StyleCraft so

much as a fact about StyleCraft *under this definition*, and Chapter 5's certified repeat-purchase findings suggest it will land somewhere near a third of the eligible file — a designed expectation the labs will confirm against the shipped answer key. That constructedness is why this chapter's evaluation sections insist on carrying the base rate everywhere: two churn models built on two defensible definitions can post identical accuracies while chasing different customers, and only the definition-plus-base-rate pair makes a reported result auditable. It is also why the platform's Churn Risk panel fails the first question before any statistics arrive: 89 percent accurate *at predicting what definition of churn* is not a rhetorical question, and a vendor who cannot answer it is selling a model of an undisclosed construct — the churn-flavored sibling of the undisclosed metric definitions that Chapter 3's one-metric-three-numbers dispute made famous.

CONCEPT

The Label Is a Policy

A churn label does more than train a model; it decides, through the model, who is eligible to be saved. Customers outside the definition — the never-activated signup, the engaged browser who stopped buying, the customer who churned faster than the window can see — are invisible to the program built on it, not because anyone decided to ignore them but because the definition decided by default. This guide's standing rule follows: the label definition appears on the deliverable's first page, with its rejected alternatives and one sentence on who each alternative would have added or removed from the treated population. The rule costs a paragraph. It converts the definition from an assumption buried in code into a policy the meeting can see, discuss, and own — which is where a decision of this size belongs.

Source: Course concept developed for this guide, informed by Neslin et al. (2006).

9.4 Logistic Regression: Probabilities from the Log-Odds Machine

The label exists; the first scorer can now be built, and this guide builds it from the instrument the reader already trusts. Section 9.2 established what the model must produce — an S-shaped probability, bounded between 0 and 1, steep in the uncertain middle and flat at the confident extremes — and logistic regression produces exactly that by the smallest possible modification of Chapter 7's machinery: keep the linear equation, change what it computes. A linear regression says the *outcome* is an intercept plus weighted features. A logistic regression says something one step removed: a quantity called the log-odds of the outcome is an intercept plus weighted features, and the probability is recovered from the log-odds by a fixed squashing curve — the logistic function — that maps any number the equation produces, however large or negative, into the interval the probability must occupy.

The intuition arrives fastest through odds, a currency marketing students already speak from every context where chances are quoted as ratios. A probability of 0.75 is odds of 3 to 1: three chances of the event for every one against. A probability of 0.5 is even odds, 1 to 1. A probability of 0.10 is odds of 1 to 9. Odds re-express probability on a scale that runs from zero to infinity instead of zero to one, and the *logarithm* of the odds runs the scale out symmetrically in both directions — negative log-odds for unlikely events, zero for even odds, positive for likely ones. That unbounded, symmetric scale is what the linear equation lives on: features move the log-odds up and down freely, the way they moved dollars in Chapter 7, and the logistic function translates the result back into a bounded probability at the end. The S-shape falls out automatically: a one-unit feature change always moves the log-odds by the same amount, but the same log-odds movement changes the probability a great deal near 0.5 and almost not at all near the extremes — which is precisely the risk behavior Section 9.2 said a straight line could not express.

DEFINITION

Logistic Function and Log-Odds

The odds of an event are the ratio of its probability to its complement's: $p \div (1 - p)$. The log-odds are the natural logarithm of the odds — an unbounded, symmetric rescaling of probability on which zero means even odds. The logistic function is the inverse translation: it converts any log-odds value back into a probability between 0 and 1, producing the characteristic S-curve that is flat near both extremes and steepest at even odds.

Source: Adapted from Hosmer et al. (2013).

DEFINITION

Logistic Regression

Logistic regression models the log-odds of a binary target as a linear function of the features — an intercept plus weighted feature values, exactly the equation form of Chapter 7 — and converts the modeled log-odds into a predicted probability through the logistic function. Its coefficients are estimated from labeled data (by maximum likelihood, machinery this guide leaves to the software), and each coefficient is read as the change in log-odds — equivalently, the multiplicative change in odds — per one-unit difference in its feature, holding the model's other features constant.

Source: Adapted from Hosmer et al. (2013) and James et al. (2021).

Reading the coefficients is where Chapter 7's disciplines transfer with one translation step, and the translation is the only genuinely new skill. A logistic coefficient lives on the log-odds scale, which no meeting speaks; exponentiating it produces an odds ratio, which careful plain language can carry. A coefficient of 0.009 on recency_days exponentiates to about 1.009: each additional day since last purchase multiplies the odds of churn by 1.009 — about nine-tenths of a percent higher odds per day, compounding, so that ninety days of additional silence multiply churn odds by roughly 2.2. A negative coefficient on app_user exponentiating to 0.55 says app users carry a little over half the churn odds of comparable non-users. Three disciplines attach immediately, all inherited. The ceteris paribus boundary of Section 7.8 applies verbatim: each statement holds the model's other features constant and is silent on everything omitted. The verb registry of Section 7.2 applies verbatim: these are associations wearing precise multiplicative clothing, and "app users churn less" earns its verb while "the app retains customers" does not — the customers chose the app, and Section 7.8's omitted-enthusiasm argument transfers without modification. And where categorical predictors do enter a model, the dummy machinery of Section 7.9 applies verbatim, with every odds ratio read *against the declared reference category* rather than absolutely. What does not transfer is any instinct to read effects on probability as constant: because of the S-curve, the same odds ratio moves a mid-risk customer's probability substantially and a low-risk customer's barely at all — statements about "percentage-point changes in churn probability" are only honest with a stated starting point, and the labs enforce the habit.

One design decision belongs here rather than in the lab, because it is a modeling judgment rather than a coding convenience. It is tempting to enter Chapter 6's named segments into this chapter's classifier as dummy variables, since the segments are memorable and the meeting speaks their names. This guide does not, for the reason Section 8.9 established: a segment assignment is a *learned* feature, produced by a scaler and a set of centroids estimated from data, so entering it as a precomputed column lets the customers being scored shape the definitions the model trains on — Table 8.2's split-contamination row, committed by a step nobody thinks of as a model. The core classifier therefore uses the behavioral variables the segments were built from, all of which the Chapter 8 feature builder already produces at the snapshot: recency, frequency, tenure, average order value, orders and revenue per month, discount share, store share, occasionwear share, and the two engagement flags. Segment membership returns

in Section 9.12's fairness appendix as an external *profiling* variable — a lens for reading who the model is treating — which is the role it can play without contaminating anything.

One practical note belongs here before the reassurances. The coefficients this section teaches you to read are coefficients in *customer units* — days, orders, shares — and the predictive candidates of Section 9.12 will not supply them, because they standardize their inputs and carry the software's default regularization, which is right for prediction and wrong for reading. Code 9.6 therefore fits a small, separate model whose only job is interpretation, labeled in the lab as ineligible for the race. Keeping the reading model and the scoring model apart is the explanation-versus-prediction boundary of Section 8.2, drawn inside a single chapter.

One reassurance and one caution close the section. The reassurance: logistic regression is this chapter's primary instrument for the same reason linear regression anchored Chapter 7 — it is transparent, stable, fast, and its coefficients can be audited against known structure, which is what Section 9.11's audit will do with the dataset's designed churn drivers (long recency, single-purchase history, discount-only buying, and store-only suburban profiles all raise churn odds by design). The caution: everything the model produces is a probability estimate conditioned on the feature-window snapshot — a statement about customers who *resembled this one*, in exactly the sense Section 8.13 established for scores, and nothing in the sharpened vocabulary of odds changes what Section 9.13 will do to the sentence "the model says she is leaving."

9.5 Trees and Neighbors: Two More Ways to Score

Logistic regression reaches a probability through an equation; this section adds two scorers that reach one through entirely different structures, and the point of the addition is only partly the methods themselves. Chapter 8's comparison discipline — same folds, same metric, same baselines, one leaderboard fixed before the race — was built for exactly this moment: multiple candidate models, structurally dissimilar, competing for one budget line. The two candidates are chosen for their instructional contrast: one scores by asking questions, the other by finding resemblances, and each fails in a way that reviews a Chapter 8 lesson from a new angle.

A classification tree scores a customer by interrogation. The model is a sequence of yes/no questions about features — is recency_days greater than 120? is frequency equal to 1? is discount_share above 0.6? — arranged so that each answer routes the customer down a branch, until she lands in a leaf: a terminal group of training customers who answered every question the same way. Her predicted churn probability is simply the churn rate among the training customers in her leaf. The tree is grown from the data by recursive splitting: at each step, the algorithm chooses the question that best separates churners from non-churners among the customers reaching that point, then repeats within each branch (Breiman et al., 1984). Two properties make trees permanently useful in marketing rooms. They are readable — a modest tree prints as a flowchart a CRM manager can walk through, and its top question is a finding in itself (on StyleCraft's designed data, expect it to be a recency split, the ninety-day rule rediscovering itself as the first branch). And they are natively interactive in Section 7.9's sense — a tree that splits on discount_share only within the low-frequency branch has found, without being told, that the discount signal matters for some customers and not others. Their characteristic failure is the one Section 8.8 built the vocabulary for: an unconstrained tree keeps splitting until leaves hold a handful of customers each, memorizing training noise leaf by leaf — overfitting in its most visible form, with the telltale gap between training and cross-validated performance of Table 8.4, controlled in practice by limiting depth or leaf size. The lab manufactures exactly this failure on purpose, entirely inside the training customers, because a tree overfitting is the fastest visual demonstration of Chapter 8's U-curve a student will ever run.

DEFINITION

Classification Tree

A classification tree predicts by routing each case through a learned sequence of yes/no feature questions to a leaf, where the predicted class probability is the share of positive-class training cases in that leaf. The tree is grown by recursive splitting — repeatedly choosing the feature question that best separates the classes among cases reaching each point — and its flexibility is governed by limits on depth or leaf size, without which it overfits in exactly the sense of Section 8.8.

Source: Adapted from Breiman et al. (1984) and Provost and Fawcett (2013).

k-nearest neighbors dispenses with fitted structure altogether. To score a customer, find the k training customers most similar to her — nearest in feature space, by the distance machinery of Section 6.6 — and predict the churn rate among those k neighbors. With k = 25, a customer whose twenty-five nearest training neighbors include ten churners is scored 0.40. The method's premise is pure resemblance: customers like you did what you will do (Cover & Hart, 1967). Its virtues are conceptual honesty — it is the "conditional average over similar customers" reading of Section 8.13 made literal — and freedom from any assumed equation shape. Its obligations and failures are equally instructive. Because it runs on distances, it inherits wholesale the scaling discipline of Section 6.7: unscaled, recency_days' hundreds dominate every 0/1 flag, and the neighbors are neighbors in recency only — the tyranny of units, third appearance, and the reason the lab's k-NN pipeline standardizes inside each training fold per the split-hygiene rule of Table 8.2. The choice of k is the bias–variance dial of Section 8.8 with a new handle: k = 1 memorizes (every training customer is her own nearest neighbor — training accuracy perfect, held-out accuracy poor), while enormous k averages everyone toward the base rate (the majority-class baseline rediscovered as a limiting case), and the lab therefore selects k by cross-validation inside the training data rather than by declaration. And k-NN produces no coefficients at all — nothing to read, nothing to audit against designed structure — which makes it this guide's standing example of the trade Table 7.4 named: predictive flexibility purchased with explanatory silence.

DEFINITION

k-Nearest Neighbors (k-NN)

k-nearest neighbors predicts a case's class probability as the positive-class share among the k training cases closest to it in feature space, under a stated distance measure computed on features scaled per Section 6.7. It fits no equation and learns no coefficients; its flexibility is governed by k, with small k overfitting and large k defaulting toward the base rate.

Source: Adapted from Cover and Hart (1967) and James et al. (2021).

Table 9.2 collects the comparison the leaderboard will run. Its last row states a designed expectation rather than a promise: on StyleCraft's data, whose churn structure was planted as a logistic relationship with a realistic noise ceiling, the three methods should land in a similar band with the logistic model competitive or ahead. That expectation is about *this dataset*, and it should not be generalized into a claim that method choice does not matter. The industry's own churn tournament found the opposite: across forty-five submissions on common validation data, differences in predictive accuracy between modeling approaches were large enough to change the profitability of a churn campaign by hundreds of thousands of dollars, with logit and tree approaches outperforming several alternatives (Neslin et al., 2006). The lesson the tournament actually teaches is the one this chapter's leaderboard enforces: approaches differ materially, so they must be compared on common validation data under a common campaign objective — which is a stronger argument for the leaderboard than the folk claim that any method will do.

Table 9.2

Three scorers compared

Dimension	Logistic regression	Classification tree	k-NN
Scores by	Equation on log-odds	Learned yes/no questions	Resemblance to k nearest training cases
Output	Probability, smooth in features	Leaf churn rates (stepwise)	Neighborhood churn rate
Readable structure	Coefficients as odds ratios	The tree itself; top splits as findings	None — no structure to read
Needs scaling (§6.7)	Not for validity	No	Yes, inside each training fold (Table 8.2)
Setting chosen by cross-validation	None required at this level	Maximum depth	k
Characteristic failure	Misses non-additive structure unless told (§7.9)	Overfits via deep growth (§8.8)	Overfits at small k; scaling neglect
Designed expectation on StyleCraft	Competitive or ahead	Close behind, readable	Close behind with honest scaling

9.6 The Confusion Matrix: Four Outcomes with Business Names

The scorers produce probabilities; suppose for the next three sections that a threshold has been chosen — the software's 0.5 will do for now, under protest that Section 9.9 will formalize — so that every customer carries a verdict. The instrument that grades verdicts is a two-by-two table so central to this chapter that its arithmetic is the chapter's verification theme: every count in it will be checked by hand in the labs, because it is the exhibit most often pasted into decks unexamined and most easily garbled in transit. Cross the truth against the prediction and every customer lands in exactly one of four cells. The model predicted churn and the customer churned: a true positive. Predicted churn and she stayed: a false positive. Predicted stay and he churned: a false negative. Predicted stay and she stayed: a true negative. The table of counts is the confusion matrix, and this guide's insistence — the reason Table 9.3 exists — is that the four cells are read in business vocabulary before any metric is computed from them, because each cell is a different event in the Comeback Edit program with a different line on the P&L.

DEFINITION

Confusion Matrix

A confusion matrix cross-tabulates actual against predicted classes at a stated threshold, partitioning all evaluated cases into true positives (predicted and actual positive), false positives (predicted positive, actually negative), false negatives (predicted negative, actually positive), and true negatives (predicted and actual negative). Every threshold-dependent classification metric — accuracy, precision, recall, and their relatives — is an arithmetic summary of these four counts, and changing the threshold changes the matrix.

Source: Adapted from Provost and Fawcett (2013) and Fawcett (2006).

Table 9.3 states the four cells in the program's own vocabulary, and its cost column is written to match the ledger Section 9.9 will derive rather than to sound expensive. Two of the four rows repay care, because the intuitive phrasing overcounts. A false negative does not cost the lost customer's entire future margin: the program could never have captured that margin with certainty, because only a fraction of treated would-churners are expected to be saved. What a false negative costs is the *missed expected incremental save* — the save rate times the retained customer's margin, which Section 9.9 will price at roughly forty dollars. A false positive does not cost the treatment plus

margin given away, either: the twelve-dollar treatment cost is already a blended figure with redemption and margin give-up inside it, so adding margin a second time double-counts. What a false positive costs is the blended twelve dollars, spent on a customer who was staying.

Table 9.3

The confusion matrix in Comeback Edit vocabulary

Cell	What happened	Program meaning	The cost, priced as Section 9.9 prices it
True positive	Offer sent; customer was leaving	A save opportunity reached	The $12 treatment cost, spent where the expected $40 save is available
False positive	Offer sent; customer was staying	A voucher sprayed at a loyal customer	The blended $12 treatment cost spent on a customer who would have stayed
False negative	No offer; customer left	A departing customer nobody tried to save	The missed expected incremental save value, estimated here at $40 — the expensive cell
True negative	No offer; customer stayed	The quiet majority, correctly left alone	Nothing — and the cell that inflates accuracy

From the four counts, the first summary metric is the obvious one: accuracy, the share of all verdicts that were correct — true positives plus true negatives, over everything. Accuracy is legitimate arithmetic, this chapter's opening vocabulary note is hereby discharged (the word that was misapplied to a numeric target in Section 8.11's vendor deck belongs here, to verdicts), and accuracy is also the most reliably misleading number in applied classification, for a reason the fourth row of Table 9.3 telegraphs: the true-negative cell is usually enormous. Churn, response, conversion, fraud — marketing's positive classes are minorities, a situation called class imbalance, and under imbalance accuracy is dominated by the model's performance on the class nobody is asking about. The arithmetic is brutal and clarifying. If thirty percent of StyleCraft's eligible customers churn, the rule "predict that nobody churns" is seventy percent accurate while identifying not one customer to save. The platform's 89 percent begins to look different against that floor — not necessarily worse, but no longer self-explanatory — and the floor has a name the reader already knows: it is the majority-class baseline, defined in Section 8.7, held at concept level through the numeric chapter with a promise that it would do serious work here. This is the work. Every accuracy figure in this chapter's deliverables is reported beside the majority-class baseline and the base rate, and an accuracy whose margin over the do-nothing rule is never computed is Chapter 8's baseline discipline waiting to deliver its verdict.

DEFINITION

Accuracy and Class Imbalance

Accuracy is the share of evaluated cases whose predicted class matches the actual class. Class imbalance is the condition — usual in marketing — in which one class heavily outnumbers the other, and under it accuracy is dominated by the majority class: the majority-class baseline of Section 8.7 achieves accuracy equal to the majority share while identifying no positive cases at all. Accuracy is therefore reported only beside the base rate and the majority-class baseline, and never as a classifier's headline under imbalance.

Source: Adapted from Provost and Fawcett (2013) and He and Garcia (2009).

Imbalance also raises a modeling question students meet the moment they search beyond this guide: an internet's worth of advice recommends rebalancing the training data — resampling the minority class upward, the

majority down, or synthesizing minority examples — so the model "sees" more churners. This guide's treatment is deliberately brief and at concept level. Rebalancing exists, sometimes helps ranking, and always distorts the model's probability scale (a model trained on a rebalanced world overestimates churn in the real one), which matters enormously in a chapter whose destination is a threshold denominated in real-world probabilities and whose Section 9.8 makes calibration an explicit deliverable. The disciplined default for problems at StyleCraft's imbalance — a minority class near a third, not a thousandth — is to change nothing about the data and everything about the evaluation: grade with the base-rate-aware instruments this chapter builds, keep every split and every fold stratified so both sides of the wall carry the true base rate, and reserve rebalancing for the rare-event regimes where it earns its distortions. The one-sentence version students should carry: imbalance is usually an evaluation problem wearing a data-problem costume.

9.7 Precision, Recall, and F1 in Campaign Terms

Accuracy failed because it pooled the four cells into one number; the repair is to ask the two questions the program actually cares about, each a different ratio of the same counts. The first question belongs to the CFO, who is paying for the treated list: *of the customers we sent the offer, how many were actually leaving?* That is precision — true positives over all predicted positives, the purity of the treated list. Its complement is the waste rate: precision of 0.5 means half the Comeback Edit budget landed on customers who were staying anyway. The second question belongs to the CRM manager, who owns the churn number: *of the customers who were actually leaving, how many did we reach?* That is recall — true positives over all actual positives, the coverage of the at-risk population. Its complement is the leak: recall of 0.67 means a third of the departing customers walked out without anyone extending a hand.

DEFINITION

Precision and Recall

Precision is the share of predicted positives that are actually positive — of those treated, how many were the real thing; the purity of the target list. Recall is the share of actual positives that were predicted positive — of the real thing, how many were caught; the coverage of the at-risk class. The two are computed from the same confusion matrix, respond to the threshold in opposite directions, and answer different stakeholders' questions; neither is "the" performance number. Precision is undefined when nothing is treated, which is why a policy table that includes a do-nothing rule must handle the empty denominator explicitly rather than divide by zero.

Source: Adapted from Provost and Fawcett (2013) and Fawcett (2006).

The two metrics are structurally in tension, and the tension is the threshold made visible. Lower the threshold and the model calls churn more freely: the treated list grows, recall rises (fewer departing customers missed), and precision falls (more loyal customers swept in). Raise the threshold and the reverse. Neither direction is progress in itself — the movement just relocates cost from one cell of Table 9.3 to the other — and this is the precise sense in which Section 9.2 insisted the score be separated from the verdict: the model fixes how good the ordering is, and the threshold chooses where along the ordering the program stops treating, which is a question about the two cells' prices, not about the model. Students should therefore hear alarm bells at any unqualified claim that a classifier "has" a precision or a recall: it has one *at a threshold*, and quoting either without the threshold — or quoting the pair achieved at different thresholds, a vendor-deck classic — is the metric-shuffling of Section 8.9 in its classification uniform.

When a single number is genuinely required — a leaderboard column, a tuning target — the convention is F1, the harmonic mean of precision and recall. The harmonic mean is the right kind of average for the job because it is dragged toward the smaller of the two: a model with precision 0.9 and recall 0.1 posts an F1 near 0.18, not the flattering 0.5 an arithmetic mean would print, so gaming one metric at the other's expense stops paying. F1's limit is stated with equal clarity: it weights the two errors equally, and this entire chapter exists because the two errors are

not equally priced. F1 is a summary for contexts that have not yet priced their errors; Section 9.9 is for contexts that have, and StyleCraft, with twelve dollars against forty, is about to become one. Table 9.4 collects this section's vocabulary as working questions — the design rule the deliverable's evaluation exhibits follow, with each metric assigned to the stakeholder whose question it answers.

Table 9.4

The working questions of classification evaluation

Metric	The question it answers	Whose question it is	When it leads the exhibit
Base rate	How common is churn under the stated definition?	Everyone; context for all else	Always present; never the headline
Accuracy	What share of all verdicts were right?	Nobody, under imbalance	Only beside base rate and majority baseline
Precision	Of those we would treat, how many are really leaving?	Finance — the budget's purity	When treatment is expensive or intrusive
Recall	Of those really leaving, how many would we reach?	CRM — the churn number's owner	When a missed positive is the dear error
F1	One balanced number, both errors weighted equally	Leaderboards and tuning	When costs are unpriced — a temporary condition
Expected net value	What does this policy earn at the stated costs?	The meeting	Whenever the two errors have been priced

9.8 Ranking Skill and Probability Quality: ROC, AUC, and Calibration

Precision and recall grade the model *after* a threshold has committed it to verdicts; this section grades the two properties of the scores themselves — how well they order customers, and whether the numbers mean what they say — before any threshold intervenes. The first instrument sweeps the threshold instead of choosing it. Start at a setting so high no one is called a churner, then lower it continuously; at each setting, record two rates: the true positive rate (recall — the share of actual churners correctly flagged) and the false positive rate (the share of actual non-churners incorrectly flagged). Plotted against each other across the full sweep, the pairs trace the ROC curve. A model with no ranking skill traces the diagonal — at every threshold it flags churners and loyalists in equal proportion, exactly what random guessing achieves. A model with perfect ranking hugs the top-left corner: all the churners flagged before any loyalist. Real models bow between the two, and the area under the curve — AUC — summarizes the bow in a single number, with an interpretation compact enough to survive a meeting: AUC is the probability that a randomly chosen actual churner receives a higher score than a randomly chosen actual non-churner, with tied scores contributing half credit (Fawcett, 2006).

DEFINITION

ROC Curve and AUC

The ROC curve plots a scorer's true positive rate against its false positive rate across all possible thresholds, tracing the trade-off between catching positives and falsely flagging negatives. AUC — the area under the ROC curve — summarizes threshold-free ranking skill and ranges from 0 to 1. An AUC of 0.5 indicates chance-level ordering; values above 0.5 indicate useful ordering in the stated direction; values below 0.5 indicate that the scoring direction is systematically reversed or worse than chance. AUC equals the probability that a randomly chosen positive case outranks a randomly chosen negative one, with ties counted as half. It is insensitive to the base rate and to any particular threshold: it grades the ordering, and neither the calibration nor the costs.

> Source: Adapted from Fawcett (2006) and Google for Developers (2026).

Three properties define AUC's proper use in this guide, each earning a paragraph of discipline. First, AUC is a good general ranking diagnostic and a poor sole criterion. It cannot be flattered by imbalance and does not depend on a threshold nobody has justified yet, which is exactly why it is the right column for a first-pass comparison of structurally different scorers. But it grades ordering across the *entire* false-positive-rate range, and the Comeback Edit decision does not live across the entire range: it lives near a particular economic threshold, inside a budgeted upper portion of the file, using probability values to compute expected economics. A model with the highest overall AUC can be inferior in the top two deciles, inferior near the 0.30 cut, worse calibrated, and worth less money. The frame therefore declares a two-stage rule rather than a single number, and the two stages answer different questions. AUC, cross-validated and compared fold by fold against a stated baseline, is the **model-family selection metric**: it chooses which kind of scorer orders customers best, and it is the right instrument for that job precisely because it is threshold-free. Expected net value at the derived threshold, computed from *calibrated* out-of-fold probabilities, is the **policy-validation gate**: the selected model must clear it, against zero and against the incumbent rule, before any money is committed. Calibration is the condition that makes the gate computable at all, and precision, recall, and treated-list size at the stated threshold are the policy diagnostics reported beside it.

The ordering of those two stages is not cosmetic. Ranking evidence is available from raw scores; economic evidence is not, because a dollar figure computed from uncalibrated probabilities is denominated in a currency the model does not print. So the family is chosen first on ordering, the chosen model is calibrated inside the training data, and only then does the arithmetic of Section 9.9 run. A leaderboard that carries a net-value column beside raw scores has quietly inverted that order, and the labs of Section 9.12 are built so that it cannot. This is Chapter 8's principle — grade the model on the artifact the meeting is choosing — carried into classification with one added clause: grade it in dollars only once the probabilities have earned the right to be multiplied by them.

Second, AUC has calibrated expectations, and the calibration is Chapter 8's too-good logic wearing this chapter's units. Individual churn behavior carries irreducible noise — the outcome window contains job changes, moves, weddings, and whims that no snapshot feature can see — and the StyleCraft data was designed with a realistic ceiling in mind, which the labs will document empirically for this population rather than assert from a general industry figure. That distinction matters: published churn AUCs vary substantially with the label, the horizon, the population, and the data available, so a number quoted as the industry's honest range certifies nothing about StyleCraft's. What transfers digit for digit is the reflex Table 8.4 installed. An AUC far above what the base's volatility, the honest baselines, and the label definition would support is not a triumph; it is a leak until proven otherwise — the label's ingredients have almost certainly contaminated the features, with the outcome-window recency of Table 8.2's third route as the usual suspect — and the auditor's first move is the feature trace, not the celebration. The labs manufacture this failure deliberately, as Chapter 8's did, because a leak a student has built is a leak she will recognize in a vendor deck.

Third, AUC is silent about the property the next section's arithmetic actually consumes. AUC grades ordering; the threshold derivation grades *levels*. A model can rank customers beautifully while producing probabilities that are systematically too high or too low — multiply every score by 0.6 and the ordering, and therefore the AUC, does not change at all, while every economic calculation built on those scores does. The property that closes the gap is calibration: a scorer is well calibrated when, among customers scored near 0.30, roughly thirty percent actually churn. Calibration is checked with a reliability exhibit — bin the scores, and compare each bin's mean predicted probability against its observed churn rate and its customer count — and, where the comparison is materially off, corrected with a cross-validated calibration procedure fitted inside the training data (scikit-learn developers, 2026a). The discipline that governs the correction is Chapter 8's: calibration is evaluated and, if necessary, repaired using training-only, out-of-fold evidence, never by adjusting a model repeatedly against the test set until the reliability plot looks agreeable.

DEFINITION

Calibration

A probability scorer is calibrated when its predicted probabilities match observed event frequencies: among cases scored near p, about a share p are positive. Calibration is distinct from discrimination — the ranking quality AUC measures — and neither implies the other. It is assessed with a reliability exhibit comparing mean predicted probability against observed positive rate within score bins, alongside each bin's count, and it is the property that entitles a threshold derived from expected costs to be applied to a model's raw scores.

Source: Adapted from scikit-learn developers (2026a).

CONCEPT

Ordering, Levels, and the Two Ways a Score Can Be Wrong

A scored file can fail its program in two independent ways, and the deliverable reports on both. It can *order* customers badly, so that the funded portion of the list is not much richer in churners than a random portion — a failure AUC, lift, and gains detect. Or it can order customers well while getting the *levels* wrong, so that a score of 0.30 does not describe a thirty-percent risk — a failure only a calibration exhibit detects, and the one that silently corrupts every dollar figure downstream, because expected value is computed from probabilities rather than from ranks.

The practical consequence is a rule about which decisions each property licenses. Good ordering alone licenses rank-and-cut programs: treat the top N, fund the top two deciles, hand the CRM team a prioritized list. Good ordering *plus* calibration is what licenses the arithmetic of Section 9.9, where a probability is multiplied by a dollar benefit and compared against a dollar cost. An analyst who applies an economically derived threshold to uncalibrated scores has not made a small technical error; they have priced a decision in a currency the model does not actually print.

Source: Course concept developed for this guide, informed by scikit-learn developers (2026a) and Provost and Fawcett (2013).

9.9 The Threshold Decision: Asymmetric Costs, Operationalized

Section 2.7 introduced the asymmetric cost of being wrong as interpretive discipline: before any analysis, name which direction of error is dearer, because the answer should shape what the analysis treats carefully. Chapter 8 previewed the asymmetry twice — in the Backstage program's two error types and in the choice between MAE and RMSE — and promised that this chapter would make it the whole instrument. This section keeps the promise, and the registry entry deserves to be honored in the text as it was written: this is the operationalization of Section 2.7. The asymmetric-cost idea stops being a lens for reading results and becomes the arithmetic that *produces* one — the decision threshold, the number that converts every score into a treatment and every error into a bill.

Begin with what the default conceals. Classification software, asked for verdicts, cuts at a probability of 0.5, and the choice feels so natural it is rarely recognized as one: call churn when churn is more likely than not (scikit-learn developers, 2026b). But 0.5 is the right threshold only under an assumption nobody at StyleCraft believes — that the two errors cost the same. Cutting at 0.5 says a wasted twelve-dollar voucher and a missed save are equivalent misfortunes, and that a customer with a 0.45 churn probability — nearly a coin flip on losing her entire future relationship — should be left alone because the coin leans slightly toward staying. The default is not a neutral setting; it is a cost assumption wearing a factory setting's camouflage, and the first act of threshold discipline is simply to see it. Two further conditions ride inside the equal-cost reading and should be stated with it: it assumes the

scores are calibrated probabilities in the sense of Section 9.8, and it assumes the stated two-error ledger contains the relevant program economics — no additional benefit or cost attached to the true positives and true negatives, no class weighting, and no transformed probability scale quietly in play.

DEFINITION

Decision Threshold

A decision threshold is the probability above which a scored case is treated as positive — here, the churn probability above which a customer receives the retention treatment. The threshold is not a property of the model: it is a policy choice that allocates the model's inevitable errors between false positives and false negatives, and it is derived from the relative costs of the two errors, not from statistical convention. The software default of 0.5 embodies the assumption that the two errors cost the same, and inherits its authority from nothing else.

Source: Adapted from Provost and Fawcett (2013) and scikit-learn developers (2026b).

The derivation replaces the default with the program's own numbers, using the expected-value framing Section 8.10 introduced at concept level. Treating a customer costs money with certainty and produces benefit only probabilistically, so the treatment is worth extending exactly when the expected benefit clears the cost. Assemble the Comeback Edit's ledger. The cost side is finance's blended figure: twelve dollars per treated customer, redemption and margin give-up included. The benefit side requires two estimates that the analyst must obtain, document, and flag as estimates: the probability that a treated would-churner is actually saved — the save rate, which finance and CRM place at roughly one in four based on past win-back performance — and the value of a save, the retained customer's expected future margin, which finance carries at roughly $160 over the planning horizon. A treated true churner is therefore worth about 0.25 × $160 = $40 in expectation; a treated loyalist returns nothing the program can claim (she was staying) and costs the full twelve dollars. The treatment rule follows in one line: treat a customer whose churn probability is p when the expected benefit p × $40 exceeds the certain cost of $12 — that is, when p exceeds 12 ÷ 40 = 0.30. The threshold is 0.30, not 0.5, and the difference is not a technical adjustment: at 0.5 the program treats only the customers the model is fairly sure about, while at 0.30 it deliberately accepts more wasted vouchers because the error the vouchers prevent is more than three times as expensive as the vouchers themselves. In other words — and this is the sentence the deliverable prints under the derivation — the threshold is where the program *chooses its mix of mistakes*, and the choice belongs to the people who pay for the mistakes, made once, in writing, in arithmetic a CFO can re-run on a napkin.

The derivation's honesty depends on four caveats, each a standing obligation. The first is calibration, and it is load-bearing rather than technical: the analytically derived threshold of 0.30 is valid only when the probabilities are sufficiently calibrated for the deployment population, because 12 ÷ 40 answers the question "at what genuine risk does the expected benefit clear the cost," and a score that is not a genuine risk cannot answer it. The deliverable therefore carries the training-only reliability exhibit beside the derivation, and where the exhibit shows material miscalibration, a training-only calibration procedure is applied before the threshold is used — never a correction chosen by watching the test set. The second is estimate uncertainty: the threshold inherits the uncertainty of its inputs, so the deliverable shows the sensitivity (at a $30 benefit the cut moves to 0.40; at $60, to 0.20), because a threshold that swings widely under plausible input changes is telling the program that its economics, not its statistics, are the binding uncertainty. The third is causal: the save rate smuggles in a claim that the offer *changes* some would-churners' behavior, which this chapter's observational machinery cannot certify, per the discipline of Section 7.2; the deliverable names the randomized holdout as the purchase that would certify it. The fourth is heterogeneity: the arithmetic assumed every customer shares one benefit figure, when Chapter 5's concentration findings guarantee they do not — a refinement (customer-specific benefit, hence customer-specific thresholds) the exercises explore and real programs eventually adopt.

BRIDGE

Chapter 11 supplies the instrument the third caveat names: a randomly held-out control inside the campaign, which is the only design that can turn the assumed save rate into a measured one.

One tension remains between this section and the budget, and resolving it is the deliverable's final piece of threshold work. The economic threshold answers "whom is it worth treating"; the budget answers "how many treatments we can afford," and nothing forces the two answers to agree. If the 0.30 cut marks 2,600 customers as worth treating and the November budget covers 1,600, the analyst faces a fork with two honest tines: cut by rank — treat the 1,600 highest scores, the capacity logic of Section 8.10's rank-and-cut, accepting that economically worthwhile treatments go unfunded — or take the gap to the meeting as a priced argument, because the lift machinery of the next section can state, in dollars, what the unfunded thousand treatments are expected to return. That argument — the model as a budget case, not just a list — is where this chapter's analyst does the most valuable work, and it is built from the exhibits the next section constructs.

CONCEPT

The Threshold Is Where the Analyst Meets the CFO

Every other number in the pipeline belongs to the analyst: the label, the split, the features, the leaderboard. The threshold belongs to the business, because it is nothing but a statement of relative prices — how many wasted treatments one missed save is worth — translated into a probability. The analyst's job is not to choose it; it is to *surface* it: to drag the cost assumption out of the software default, price the two errors with the people who own their budgets, derive the cut as arithmetic, show its sensitivity, evidence the calibration that entitles it to be applied, and record the decision.

An analyst who does this once changes how the room thinks about every scored program it funds afterward, because the question "what threshold, and from what costs?" — once heard — is impossible to unhear. This is the chapter's verification theme in its managerial form: any threshold you are handed, including your own software's, must be re-derivable from stated costs, and a threshold that cannot be is a cost assumption nobody agreed to.

Source: Course concept developed for this guide, informed by Provost and Fawcett (2013).

9.10 Propensity, Deciles, and the Lift and Gains Exhibits

The threshold converts scores to verdicts one customer at a time; this section builds the exhibits that convert the *whole scored file* into the meeting's language, and it starts by giving the score its industry name and disambiguating it. A behavioral propensity score is a model-estimated probability that a customer will exhibit a behavior — churn, respond, convert, upgrade — used to rank customers for differential treatment. The term is worth owning because it is the form in which this chapter's machinery is packaged everywhere in the marketing stack: every CDP "likelihood to buy," every CRM "response score," every retention flag is a propensity score with a wardrobe, and the questions this chapter has equipped — what label, what window, what leakage audit, what calibration, what threshold, against what baseline — are the questions that audit them all.

The disambiguation matters because the phrase has a second, established meaning that Chapter 11 will need. In causal inference, a propensity score is the estimated probability of receiving a *treatment* conditional on covariates, used to adjust comparisons between treated and untreated groups. This chapter uses *behavioral propensity score* throughout to mean a predicted probability of a marketing behavior. It is not the treatment-assignment propensity score of causal inference, and the two should never be substituted for one another in a sentence: one ranks customers for an offer, and the other tries to make an observational comparison behave more like an experiment.

DEFINITION

Behavioral Propensity Score and Ranked Targeting

A behavioral propensity score is a model-estimated probability of a defined customer behavior, used to rank customers for treatment. Ranked targeting treats customers in descending score order — to a threshold, a budget, or a capacity — so that the program's yield depends on the ordering's quality rather than on any individual score being exactly right. Its standard evaluation exhibits are the decile table, the lift chart, and the gains chart, all built by sorting the evaluated file by score and comparing capture against the base rate. The term is distinct from the treatment-assignment propensity score of causal inference, which estimates the probability of *receiving* a treatment and is introduced in Chapter 11.

Source: Adapted from Provost and Fawcett (2013) and Neslin et al. (2006).

The exhibits are built by one mechanical act — sort the evaluated customers by score, highest first, and slice the sorted file into ten equal deciles — followed by bookkeeping. For each decile, count the actual churners it contains and compute two quantities. Lift is the decile's churn concentration relative to the base rate: if the top decile's customers churn at 2.4 times the overall rate, the model's best ten percent is 2.4 times richer in the target class than a randomly chosen ten percent — a number with a built-in baseline, which is why it survives in industry vocabulary. Cumulative gains is the running capture: treating the top decile reaches what share of all churners; the top two deciles, what share; and so on to the whole file, where capture reaches one hundred percent by definition. Plotted, the gains curve rises steeply while the model's ordering is finding churners and flattens as the remaining file thins out, against the diagonal that random targeting would trace — the visual sibling of the ROC curve, redrawn in the units a budget meeting thinks in: how much of the problem do we reach for how much of the file.

The exhibits earn their central place because they price program designs directly, and the reader has seen this move before: it is the capture head-to-head of Section 8.10, matured into a full curve. If the top two deciles of StyleCraft's scored file capture on the order of half of all churners — the designed expectation for a model near the data's quality ceiling, which the lab will test — then the "treat the top two deciles" program reaches roughly half the at-risk margin for a fifth of the blanket program's cost, and the comparison against the alternatives is finally on one page in one currency. The ninety-day rule occupies a fixed point on the same axes (its treated list is one particular slice of the file, with its own capture), the blanket occupies the far corner (all cost, all capture), and the marginal logic of where to stop is read straight off the curve's slope: keep funding deciles while the incremental capture in a decile is worth more than the decile costs to treat — the threshold economics of Section 9.9 re-expressed as a stopping rule on a chart, which is how it most often survives contact with a budget meeting. The decile table, the lift chart, and the gains chart are, for ranked-treatment programs, what the leaderboard was for model choice in Chapter 8: the fixed exhibit family within which every candidate — model, rule, and blanket — competes on the same field, and this chapter's deliverable closes on them.

One reading discipline attaches to the decile table, because the draft version of this sentence is easy to overstate. Lift should generally be strongest in the early deciles and weaken as the list deepens; that is what a model with real ordering skill does. Perfect monotonic decline is not guaranteed in a finite evaluation sample, and a non-monotone middle is a finding about where the ordering blurs rather than an error to hunt down. What *would* be an error is a top decile whose lift sits near 1.0, because that says the model's best ten percent is no richer in churners than a random ten percent — the ranking has failed at exactly the end of the file the program funds.

9.11 AI as a Classification Assistant

The division of labor this guide has developed across three chapters — the assistant drafts mechanics, the analyst audits meaning — reaches its most consequential form here, because classification is where an assistant's unexamined defaults stop being statistical choices and start being business policy. A current assistant, handed the feature table and "build a churn model," will construct the label, fit three model families, tune them, print a

confusion matrix, and report accuracy — and in that single fluent response it will have chosen a churn definition (undisclosed), a threshold (0.5, unmentioned), and a headline metric (the one this chapter spent Section 9.6 dethroning), while narrating the result with congratulations. None of these is a bug; each is a default standing in for a decision nobody was asked to make. The failure modes deserve their names, because the audit targets them.

The accuracy celebration: the assistant reports "91 percent accuracy" without the base rate or the majority-class baseline, and under imbalance the margin over the do-nothing rule is left uncomputed — the narration inherits the internet's enthusiasm for accuracy exactly as Chapter 8's assistants inherited its enthusiasm for R^2. The silent threshold: the confusion matrix arrives cut at 0.5, no costs discussed, and every downstream number — precision, recall, the treated-list size — quietly inherits the unexamined default; asked for "the model's precision," the assistant answers as if the threshold were a fact of nature. The label improvisation: asked for a churn model without a definition, the assistant invents one — often "no purchase in the last N days" computed *at the analysis date*, which builds the label out of the feature window and manufactures the outcome-window contamination of Table 8.2 in a single step; the resulting AUC is spectacular, and spectacularly meaningless. The test-set race: three models fitted and compared on test AUC, then a depth or a k chosen by watching the same test numbers move — the sealed set spent as a development instrument, which is Section 8.5's violation in its classification uniform. The rebalancing reflex: prompted with "imbalanced data," many assistants reach for resampling by default, distorting the probability scale that Section 9.9's threshold arithmetic consumes — a repair applied to a problem the evaluation should have absorbed. The uncalibrated economics: expected-value tables computed from raw scores with no reliability exhibit anywhere, so the dollars are denominated in probabilities nobody checked. And the verdict language: scores narrated as facts — "the model identified 412 customers who will churn" — the probability-to-verdict collapse of Section 9.2, performed by a sentence.

Where the assistant genuinely helps, use it deliberately, and the list is long because the mechanics are real work: the label-construction code from a written definition (the definition supplied, never delegated); the split, the stratified fold generator, the scalers inside the training folds, and the three-model scaffold from the declared frame; the confusion-matrix, reliability, decile, lift, and gains boilerplate, which is tedious and perfectly specifiable; error analysis — "which actual churners did the model score lowest, and what do they share?" surfaces the lumpy-cadence miss faster than any manual pass; and threshold sensitivity tables from stated cost ranges, which is exactly the well-specified arithmetic assistants execute reliably. The governing instrument extends Chapter 8's audit rather than replacing it: the five points of Table 8.5 run unchanged — frame, leakage, split hygiene, baseline, error in decision units — and Table 9.5 adds the four checks that classification makes necessary. Together they are this chapter's verification theme made procedural: the confusion matrix is re-added by hand, and the threshold is re-derived from stated costs, before any result is believed or repeated.

Table 9.5

The classification supplement to the five-point audit (run with Table 8.5)

Supplemental point	The check	Fails when
S1. Base rate and floor	Base rate stated; majority-class baseline computed on the same evaluation set; accuracy read only against both, with the margin computed	Accuracy is quoted alone, or the do-nothing rule was never priced
S2. Matrix arithmetic	The confusion matrix's four counts re-added by hand: cells sum to the evaluation set; precision and recall recomputed from raw counts	Any count fails to reconcile, or metrics arrive without their matrix
S3. Threshold provenance	The threshold identified, its cost assumption stated, and the cut re-derived from the program's own FP/FN economics	The threshold is 0.5 by inheritance, or cannot be re-derived from any stated costs
S4. Calibration and selection hygiene	Model family and settings chosen by cross-validated ranking evidence inside	A net-value column sits beside raw scores; the calibration check covers a

Supplemental point	The check	Fails when
	the training customers; the selected model calibrated inside the training data; every expected-value figure computed from out-of-fold *calibrated* probabilities; the test set opened once	model other than the one selected; or a depth, a k, or a family was chosen by watching test performance

AI IN PRACTICE

The Classifier That Must Survive Its Audit

The two-prompt pattern of Sections 7.11 and 8.11, with the supplement between. Prompt one, the frame as specification: paste the churn definition of Section 9.3 verbatim — snapshot, windows, activity rule, both eligibility conditions — plus the feature list with derivation rules and the snapshot-only instruction, the split protocol with seed and stratification, the declared selection protocol (stratified five-fold cross-validation inside the training customers), and the baselines named (a constant-score floor, a recency-only logistic baseline, and the ninety-day rule evaluated as a fixed policy). Then give the instruction that ends the improvisation: "Build exactly this pipeline. Select the model family and every setting using stratified five-fold cross-validation on the training customers only. Report one leaderboard graded on cross-validated AUC alone — no dollar figures beside raw scores. Then calibrate only the selected model, inside the training data, and report its out-of-fold reliability table and its cross-validated expected net value at the stated threshold. Do not touch the test set. Output scores, not verdicts. Choose no threshold yourself. Narrate nothing yet."

Then run Table 8.5's five points, followed by Table 9.5's four: confirm the label came back as defined and was built from purchase activity rather than net revenue; trace three features across the snapshot and run the recency range check of Section 8.12; confirm the scalers lived inside the folds and no test customer was consulted; read the cross-validated AUCs against the base's documented volatility, treating anything implausibly high as a leak to trace rather than a result to enjoy; check that the reliability exhibit describes the model that was actually selected and not some other candidate, and that every dollar figure in the reply descends from those calibrated probabilities; then — and only then — open the test set once, apply the threshold *you* derived in the Section 9.9 arithmetic, produce the confusion matrix, and re-add its four cells by hand against the test-set count. Prompt two, after survival: "Draft the recommendation paragraph for a marketing VP: the model's capture in the top two deciles against the ninety-day rule's, the treated-list size at the 0.30 threshold with the cost assumptions stated, and no sentence that asserts an individual customer will churn." Audit the draft's verbs per Section 7.2 — propensity language earns "is likely to," never "will" — file both exchanges per Appendix D, and sign the scored list as the analyst of record, threshold and all.

Source: Course concept developed for this guide, informed by scikit-learn developers (2026c) and Provost and Fawcett (2013).

9.12 Hands-On Application in Python and Google Colab

The preceding sections built the vocabulary; this section spends it on the Comeback Edit decision, in two labs that mirror the chapter's argument. Lab 9.1 makes every number touchable: the confusion matrix and the five-policy economics by hand at miniature scale, then the frame, the label, the base rate, and the sealed split at full scale, and finally an interpretive logistic model whose coefficients can be read in customer units. Lab 9.2 runs the race and carries it to the decision, in the order Section 9.8 declared: flexibility chosen inside the training folds by a one-

standard-error rule, a five-candidate leaderboard graded on ranking alone, the finalist calibrated inside the training data and put through an economic gate, the sealed test set opened once for the frozen model, the five competing policies priced on one table, the ROC, lift, and gains exhibits plotted, and the fairness appendix computed group by group.

The labs reuse Chapter 8's frame wholesale, and the reuse is made explicit in code rather than assumed: same snapshot of December 31, 2025, same feature builder from Codes 8.2 and 8.3, same outcome window, same two-condition eligibility rule, same seed. Two conventions hold throughout, and both are Chapter 8's disciplines made structural. Every model, including every baseline, is an estimator evaluated through the same cross-validation call on the same stratified folds, so that "same folds, same metric, same baselines" is enforced by the code rather than remembered by the analyst. And every step that learns anything from data — a standardizer, the estimator itself, a calibrator — lives inside a pipeline, so that it is refitted within each training fold and never estimates a quantity from a customer it is about to be graded on (scikit-learn developers, 2026c). AI assistants may draft any code cell (Appendix C has templates; Appendix A covers Colab mechanics); every output is predicted before it is computed, and every exchange is documented per Appendix D.

9.12.1 Lab 9.1, Part A: The Confusion Matrix and the Ledger in Miniature

The miniature is the ten-customer table one last time, now carrying everything the running example has earned: the outcome-window activity from Lab 8.1 determines who actually churned (C002, C006, and C007), and a hypothetical logistic model, fitted to customers like these, has scored each customer's churn probability. The scores are chosen to be consistent with the designed churn drivers — long silence scores high, steady cadence scores low — and to contain the chapter's lessons in ten rows. Before running anything, predict which customers the model will get wrong at a threshold of 0.5, and in which direction.

Code 9.1. Verify two confusion matrices by hand

```
import pandas as pd

mini = pd.DataFrame({
    "customer_id": ["C001","C002","C003","C004","C005",
                    "C006","C007","C008","C009","C010"],
    "churn_prob": [0.05, 0.35, 0.15, 0.70, 0.10,
                   0.90, 0.60, 0.20, 0.65, 0.45],
    "churned":    [0, 1, 0, 0, 0, 1, 1, 0, 0, 0],
    # The incumbent's own column: no purchase in the 90 days
    # before the snapshot. A fixed policy, not a score.
    "lapsed_90d": [0, 0, 0, 1, 0, 1, 0, 0, 1, 1],
}).set_index("customer_id")

for t in (0.50, 0.30):
    pred = (mini["churn_prob"] >= t).astype(int)
    tp = int(((pred == 1) & (mini["churned"] == 1)).sum())
    fp = int(((pred == 1) & (mini["churned"] == 0)).sum())
    fn = int(((pred == 0) & (mini["churned"] == 1)).sum())
    tn = int(((pred == 0) & (mini["churned"] == 0)).sum())
    assert tp + fp + fn + tn == len(mini), "the matrix must sum"
    prec = tp / (tp + fp) if (tp + fp) else float("nan")
    rec  = tp / (tp + fn) if (tp + fn) else float("nan")
    f1   = 2 * prec * rec / (prec + rec) if (prec + rec) else float("nan")
    print(f"threshold {t:.2f}: TP {tp}  FP {fp}  FN {fn}  TN {tn} | "
          f"acc {(tp + tn) / len(mini):.2f}  prec {prec:.2f}  "
          f"rec {rec:.2f}  F1 {f1:.2f}")
```

Expected output: two lines, one per threshold, each carrying the four cell counts and the four metrics computed from them; one assertion per threshold passes silently.

Input: ten literal customers with a score, a truth, and the incumbent's flag. Transformation: two thresholds applied to one set of scores. Output: the two confusion matrices whose every figure the Verification Check re-

derives by hand. The assertion is the smallest useful habit in the chapter: a matrix that does not sum to the evaluated population has lost customers somewhere, and a matrix that has lost customers is a matrix whose metrics mean nothing.

VERIFICATION CHECK

Before you run: build both matrices by hand, customer by customer. At the 0.50 threshold the model flags C004, C006, C007, and C009. True positives: C006 and C007. False positives: C004 and C009 — the two customers who ignite from nothing, whom the model (reading their long silence) reasonably condemned and who bought anyway. False negative: C002 — the occasion shopper whose big November baskets gave her a recent purchase and a comfortable 0.35 score, and who never came back: the lumpy-cadence miss, Chapter 5's finding wearing its third costume. True negatives: the remaining five. Accuracy is (2 + 5) ÷ 10 = 0.70; precision 2 ÷ 4 = 0.50; recall 2 ÷ 3 ≈ 0.67; F1 = 2(0.50)(0.67) ÷ (0.50 + 0.67) ≈ 0.57. At the 0.30 threshold — the economic cut Section 9.9 derives — the flag list grows to six as C002 and C010 join: TP 3, FP 3, FN 0, TN 4; accuracy 0.70; precision 0.50; recall 1.00; F1 ≈ 0.67.

After you run: reconcile every printed figure against your hand arithmetic, then notice the result the next cell will price. The majority-class baseline — treat nobody — is 7 ÷ 10 = 0.70 accurate. The model at either threshold is *also* 0.70 accurate. Three policies, one accuracy, and nothing in that number distinguishes them. Write the sentence in your notebook before continuing: accuracy has just failed to detect a difference the program will pay for.

Investigate if: your counts disagree with the printed ones anywhere. Every number in this cell is reproducible with a pencil, and a mismatch is arithmetic rather than convention.

The miniature's payoff is not the matrices; it is what happens when the four counts meet the ledger. Code 9.2 prices five policies on the same ten customers, using the Section 9.9 figures — twelve dollars per treatment, forty dollars of expected benefit per treated true churner — and it handles the empty denominator explicitly, because a do-nothing policy treats nobody and precision is undefined rather than zero.

Code 9.2. Price five policies on one ledger

```
COST, BENEFIT = 12.0, 40.0   # per treatment; per treated true churner

def price(name, treat, truth=mini["churned"]):
    """Confusion counts and expected net value for any treatment vector.
    Precision is undefined -- not zero -- when nothing is treated."""
    treat = pd.Series(treat, index=truth.index).astype(int)
    tp = int(((treat == 1) & (truth == 1)).sum())
    fp = int(((treat == 1) & (truth == 0)).sum())
    fn = int(((treat == 0) & (truth == 1)).sum())
    tn = int(((treat == 0) & (truth == 0)).sum())
    treated = tp + fp
    return {"policy": name, "TP": tp, "FP": fp, "FN": fn, "TN": tn,
            "treated": treated,
            "accuracy": (tp + tn) / len(truth),
            "precision": tp / treated if treated else float("nan"),
            "recall": tp / (tp + fn) if (tp + fn) else float("nan"),
            "net": tp * BENEFIT - treated * COST}

policies = pd.DataFrame([
    price("treat nobody",        0),
    price("model at 0.50",  (mini["churn_prob"] >= 0.50).astype(int)),
    price("model at 0.30",  (mini["churn_prob"] >= 0.30).astype(int)),
```

```
    price("ninety-day rule",     mini["lapsed_90d"]),
    price("treat everyone",      1),
]).set_index("policy")

print(policies.round(2).to_string())
```

Expected output: one five-row table carrying the four confusion counts, the treated count, accuracy, precision, recall, and expected net value for every policy — with precision printed as NaN for the do-nothing row rather than raising a division error.

Input: the miniature's scores, truths, and incumbent flag. Transformation: five treatment vectors run through one pricing function. Output: the exhibit the whole chapter turns on, at a scale where every cell is checkable by hand. Writing the pricing as one function rather than five blocks is the point of the cell's construction: every policy is graded by identical arithmetic, which is the miniature version of Section 8.9's same-metric clause.

VERIFICATION CHECK

Before you run: price all five policies with a pencil. Treat nobody: nothing spent, nothing saved, net $0. The 0.50 cut treats four customers for $48 and reaches two churners worth $80, netting +$32. The 0.30 cut treats six for $72 and reaches all three for $120, netting +$48. The ninety-day rule treats C004, C006, C009, and C010 — four customers for $48 — and catches one churner worth $40, netting −$8. Treat everyone treats ten for $120 and reaches all three for $120, netting exactly $0 — the CFO's instinct, confirmed to the dollar.

After you run: reconcile all five nets, then write the two sentences the miniature has earned. First, accuracy cannot distinguish policies whose economics differ by forty-eight dollars on ten customers: three rows post exactly 0.70 while netting $0, +$32, and +$48, and across all five policies the spread runs from −$8 to +$48 with no help from the accuracy column. Second, the threshold — not the model — is what moved the program from +$32 to +$48, because the same scores were cut in a different place. Then identify the single customer whose reclassification between the two thresholds contributes most to the improvement, and say in one sentence what her profile implies about which customers live near a threshold.

Investigate if: the ninety-day rule's net is not negative. On these ten customers the incumbent treats three loyal customers and one churner, which is exactly the failure Section 9.1 predicted from Chapter 5's lumpy cadence — and a positive net means the flag column was transcribed incorrectly.

9.12.2 Lab 9.1, Part B: The Frame, the Label, and the Sealed Split

Part B builds the real thing on Chapter 8's frame. Begin, per the standing rule, with the frame in a markdown cell: the churn definition of Section 9.3 in its four-element form, both eligibility conditions, the declared selection protocol (stratified five-fold cross-validation inside the training customers), the declared baselines (constant score, continuous recency, and the ninety-day rule as a fixed policy), and the split protocol with its seed. Only then does code run. Code 9.3 rebuilds the snapshot feature table using the same parameterized builder Chapter 8 wrote, so that nothing in this chapter's pipeline depends on a notebook the reader may not have open.

Code 9.3. Rebuild the snapshot frame from Chapter 8

```
import numpy as np
import pandas as pd

# The companion repository ships Chapter 8's Codes 8.2 and 8.3 as a
# module. Students who prefer to read the source can paste those two
# cells instead; the columns are identical either way.
```

```
from stylecraft_features import build_customer_features

FEATURE_START = pd.Timestamp("2024-07-01")
SNAPSHOT      = pd.Timestamp("2025-12-31")
OUTCOME_START = pd.Timestamp("2026-01-01")
OUTCOME_END   = pd.Timestamp("2026-06-30")

# Certified files are loaded per Appendix A as transactions_clean,
# customers, products, and stores.
cf_snap = build_customer_features(transactions_clean, customers, products,
                                  snapshot=SNAPSHOT,
                                  window_start=FEATURE_START)

tx = transactions_clean.assign(
    order_date=transactions_clean["order_date"].dt.normalize())
tx_feat = tx[tx["order_date"].between(FEATURE_START, SNAPSHOT,
                                      inclusive="both")]
tx_out  = tx[tx["order_date"].between(OUTCOME_START, OUTCOME_END,
                                      inclusive="both")]

# Both ends of both windows, asserted rather than assumed (Section 8.3).
assert tx_feat["order_date"].min() >= FEATURE_START
assert tx_feat["order_date"].max() <= SNAPSHOT
assert tx_out["order_date"].min()  >= OUTCOME_START
assert tx_out["order_date"].max()  <= OUTCOME_END

# ELIGIBILITY, both conditions evaluable at the snapshot (Section 9.3).
signed_up = set(customers.loc[customers["signup_date"] <= SNAPSHOT,
                              "customer_id"])
purchased = set(cf_snap.index)
eligible  = sorted(signed_up & purchased)
model_df  = cf_snap.loc[eligible].copy()

print("customers in the file:                    ", len(customers))
print("signed up by the snapshot:                ", len(signed_up))
print("purchased inside the feature window:      ", len(purchased))
print("eligible (both conditions) -> model_df:   ", len(model_df))
print("signed up, never purchased (activation):  ",
      len(signed_up - purchased))
print("acquired after the snapshot (out of scope):",
      len(customers) - len(signed_up))
```

Expected output: six counts that reconcile against the customer file, and four window assertions passing silently.

Input: the certified transaction, customer, product, and store files. Transformation: one parameterized feature build at the snapshot, two window filters with both ends enforced, and a two-condition eligibility intersection. Output: model_df, one row per eligible customer, carrying only snapshot-safe features — and a printed partition of the customer file that names the excluded groups rather than dropping them silently. The activation population in the fifth line is the group Section 9.3 removed on purpose: customers who signed up and never bought, for whom recency, average order value, discount share, and store share are all undefined. They are a real marketing problem and a different one.

Code 9.4. Construct the churn label from purchase activity

```
# The label follows the DEFINITION -- no completed purchase order in the
# outcome window -- not the convenience of a zero revenue total.
outcome_orders = tx_out.groupby("customer_id")["order_id"].nunique()
model_df["outcome_orders"] = outcome_orders.reindex(model_df.index,
                                                    fill_value=0)
model_df["churned"] = model_df["outcome_orders"].eq(0).astype(int)
```

```
# Chapter 8's spend label, kept for reconciliation ONLY. It is not the
# label and it is never a feature.
outcome_spend = tx_out.groupby("customer_id")["line_revenue"].sum()
model_df["spend_next6m"] = outcome_spend.reindex(model_df.index,
                                                 fill_value=0.0)

print("eligible customers:", len(model_df))
print("base rate (churned):", round(model_df["churned"].mean(), 3))

# Reconciliation 1: activity vs. net revenue. These need NOT agree.
zero_spend = model_df["spend_next6m"].le(0)
disagree = int((zero_spend != model_df["churned"].astype(bool)).sum())
print("customers where zero net revenue and zero purchase activity "
      f"disagree: {disagree}")
if disagree:
    print(model_df.loc[zero_spend != model_df["churned"].astype(bool),
                       ["outcome_orders", "spend_next6m"]].head())

# Reconciliation 2: the shipped answer key (dataset spec, Section 4).
cf_key = pd.read_csv("customer_features.csv").set_index("customer_id")
agreement = model_df["churned"].eq(
    cf_key["churn_label"].reindex(model_df.index)).mean()
print("agreement with the shipped label:", round(agreement, 4))
```

Expected output: the eligible count, the base rate, the count of customers whose activity and revenue definitions disagree (with a sample of them if any exist), and the agreement rate against the shipped answer key.

Input: the outcome-window transactions and the eligible feature table. Transformation: a distinct-order count per customer, reindexed onto the eligible population so that a customer with no outcome-window rows receives zero rather than a missing value. Output: the churn flag, plus two reconciliations the deliverable reports. The reindex-with-fill is the cell's load-bearing line, and it is the classification twin of Chapter 8's fillna argument: a customer absent from the outcome-window groupby is not missing data, she is the positive class.

VERIFICATION CHECK

Before you run: write three predictions. The base rate — from Chapter 5's certified repeat-purchase findings, predict the share of eligible customers with no outcome-window purchase; the designed expectation is in the neighborhood of a third, and a printed value far from your prediction is a label bug until explained. The disagreement count between the activity definition and a net-revenue definition — predict whether it is zero, and write down what a non-zero count would tell you about the revenue column. And the dependency: confirm in writing that the label was built from outcome-window data alone and every feature from feature-window data alone, the wall of Section 8.3 restated once per chapter because every leak in this chapter's history walked through it.

After you run: grade all three. If the disagreement count is non-zero, open the sampled rows and classify each: a zero-value order, a fully returned order, or a negative net. Then write the sentence the deliverable needs on its first page — whether, on this certified file, the churn flag happens to coincide with the complement of Chapter 8's spend label, and why the chapter still refuses to define it that way. If agreement with the shipped key is not exactly 1.0, find a disagreeing customer and determine which definitional lever of Table 9.1 the two labels moved differently; that forensic exercise is worth more than the agreement figure.

Investigate if: the base rate lands above one half. Under this six-month definition it should not, and a majority-churn file usually means the outcome window was filtered incorrectly or the eligibility intersection was skipped, admitting never-activated signups who are churned by construction.

Code 9.5. Split the eligible population and seal the test set

```
from sklearn.model_selection import train_test_split

# Chapter 8's snapshot-safe columns, minus monetary: by Table 6.5's own
# derivation rules monetary equals aov times frequency, so including all
# three walks straight into Section 7.10's multicollinearity trap.
FEATURES = ["recency_days", "frequency", "tenure_days", "aov",
            "orders_per_month", "revenue_per_month", "discount_share",
            "store_share", "occasionwear_share", "app_user",
            "email_opt_in"]

assert set(FEATURES) <= set(model_df.columns), "a feature does not exist"
for forbidden in ["churned", "outcome_orders", "spend_next6m"]:
    assert forbidden not in FEATURES, f"{forbidden} was handed over"

train_df, test_df = train_test_split(
    model_df, test_size=0.20, random_state=42,
    stratify=model_df["churned"])

train_ids, test_ids = train_df.index, test_df.index
assert train_ids.intersection(test_ids).empty
assert len(train_ids) + len(test_ids) == len(model_df)

X_tr, y_tr = train_df[FEATURES], train_df["churned"]

# Development diagnostics come from the TRAINING customers only. The test
# base rate is deliberately NOT printed: knowing it would leak forward
# into every expectation about what a plausible result looks like.
print("training customers:", len(train_ids))
print("test customers:    ", len(test_ids))
print("training base rate:", round(y_tr.mean(), 3))
print("training majority-class accuracy:",
      round(max(y_tr.mean(), 1 - y_tr.mean()), 3))
print("\nTEST SET SEALED. Its outcomes stay unopened until Code 9.14.")
```

Expected output: two population counts, the training base rate and its majority-class floor, and the sealing notice; four structural assertions pass silently.

Input: the labeled eligible table. Transformation: a seeded, stratified 80/20 split at the customer grain. Output: two frames and two index objects that every later cell reuses. Two details enforce disciplines this chapter argued for. The stratify argument is Section 9.6's one-line concept-level answer to imbalance: both sides of the split inherit the true base rate, every fold inherits it too, and nothing about the probability scale is distorted the way rebalancing would distort it. And the cell prints the *training* base rate and floor rather than the test ones, because a sealed test set is not sealed once its outcomes have been seen; the test figures appear in Code 9.14, after the grade is recorded, where the same numbers are context for reading a result rather than an influence on producing one.

9.12.3 Lab 9.1, Part C: Reading the Risk in Odds

Part C reads a model before any model is raced, because Section 9.4's odds-ratio skill needs a model whose coefficients are in customer units and it will not get one from the leaderboard. The candidates of Lab 9.2 standardize their inputs and carry scikit-learn's default regularization, which is right for prediction and wrong for reading: a coefficient on standardized recency is a statement about standard deviations, and a penalized coefficient is shrunk toward zero by an amount nobody in the meeting agreed to. Code 9.6 therefore fits a separate, deliberately small model whose only job is interpretation — the explanation-versus-prediction boundary of Section 8.2, drawn inside one chapter's lab rather than across two.

Code 9.6. Fit and read an interpretive logistic model

```
import statsmodels.formula.api as smf

# AN INTERPRETATION EXERCISE, NOT A CANDIDATE. This model is fitted on
# the training customers in original units and is never scored, never
# entered on the leaderboard, and never used to treat anyone. Its job is
# to make Section 9.4's odds-ratio sentences checkable.
interpretive = smf.logit(
    "churned ~ recency_days + frequency + tenure_days"
    " + app_user + email_opt_in + discount_share + store_share",
    data=train_df).fit(disp=False)

print(interpretive.summary().tables[1])

odds = np.exp(interpretive.params).rename("odds_ratio")
ci = np.exp(interpretive.conf_int()).rename(columns={0: "lo", 1: "hi"})
print("\n" + pd.concat([odds, ci], axis=1).round(3).to_string())

b = interpretive.params["recency_days"]
print(f"\nper additional day of silence, churn odds x {np.exp(b):.4f}")
print(f"over 90 additional days,        churn odds x {np.exp(90*b):.2f}")
```

Expected output: the coefficient table with standard errors, z statistics, and p-values; a second table giving each exponentiated coefficient with its confidence interval as an odds ratio; and the recency effect translated to one day and to ninety days.

Input: the training customers only. Transformation: one maximum-likelihood fit on seven predictors in their native units. Output: odds ratios a meeting can hear. Three design choices carry the section's argument. The feature list is shorter than the predictive one, because a model built to be read should contain only variables whose coefficients someone is prepared to defend. The fit is on training customers rather than on everything, because a chapter that seals its test set does not open it for a side exercise. And the model is labeled in the comment as ineligible for the race, because the fastest way to lose the explanation-versus-prediction boundary is to let one artifact quietly play both roles.

VERIFICATION CHECK

Before you run: write the predicted sign of every coefficient from the designed churn structure and Chapter 5's findings — recency positive (silence raises churn odds), frequency negative, tenure negative, the two engagement flags negative, discount share positive, store share positive. Then predict, to one decimal place, the odds multiplier for ninety additional days of silence.

After you run: grade every sign, then translate two coefficients into plain odds language per Section 9.4 — one continuous ("each additional day of silence multiplies churn odds by …") and one flag ("app users carry … times the churn odds of comparable non-users") — and check each sentence's verb against Section 7.2's registry. Add the ceteris paribus clause explicitly, and then write the sentence that limits it: because of the S-curve, the same odds ratio moves a mid-risk customer's probability substantially and a low-risk customer's barely at all, so any statement about percentage points needs a stated starting probability.

Investigate if: a sign contradicts the designed structure, or a confidence interval for an odds ratio comfortably straddles 1.0 for a variable the design says matters. The first is a data or formula problem worth tracing; the second is honest evidence of a weak effect, and the deliverable should report it as such rather than quoting the point estimate alone.

9.12.4 Lab 9.2, Part A: Choosing Flexibility Inside the Training Data

Chapter 8's Part B made the U-shape visible without consulting the test set, and this lab owes the same discipline to two settings this chapter introduced: the tree's maximum depth and k-NN's k. Both are chosen here, from training-only evidence, with cross-validated AUC as the criterion and training AUC printed beside it so the overfitting signature of Table 8.4 is visible rather than inferred. The selection is not a bare maximum. Section 8.8 argued that flexibility must earn its place, and Code 9.7 makes that argument executable with a one-standard-error rule: find the best mean cross-validated AUC, define an acceptable band one standard error below it, and inside that band take the *simplest* candidate — the shallowest tree, and the largest k, since a larger neighborhood is a smoother and less flexible model.

Code 9.7. Select the tree's depth inside the training folds

```
from sklearn.model_selection import StratifiedKFold, cross_validate
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.metrics import roc_auc_score

# ONE fold generator, built once, shared by every candidate and every
# selection decision in this lab: the same-split clause of Section 8.9.
CV = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

def auc_scorer(estimator, X, y):
    return roc_auc_score(y, estimator.predict_proba(X)[:, 1])

def one_se_band(table):
    """Rows whose mean CV AUC is within one standard error of the best.
    This is what turns 'prefer the simpler model when two are
    indistinguishable' from advice into an algorithm."""
    best = table["cv_auc"].idxmax()
    se = table.loc[best, "cv_sd"] / np.sqrt(CV.get_n_splits())
    return table[table["cv_auc"] >= table.loc[best, "cv_auc"] - se]

rows = []
for depth in [2, 3, 4, 5, 6, 8, 12, 0]:      # 0 means "no depth limit"
    md = None if depth == 0 else depth
    tree = DecisionTreeClassifier(max_depth=md, random_state=42)
    s = cross_validate(tree, X_tr, y_tr, cv=CV,
                       scoring={"auc": auc_scorer},
                       return_train_score=True)
    rows.append({"max_depth": "none" if md is None else str(md),
                 "complexity": 99 if md is None else md,
                 "train_auc": s["train_auc"].mean(),
                 "cv_auc": s["test_auc"].mean(),
                 "cv_sd": s["test_auc"].std(),
                 "gap": s["train_auc"].mean() - s["test_auc"].mean()})
tree_flex = pd.DataFrame(rows).set_index("max_depth")
print(tree_flex.round(3).to_string())

# Simplest candidate inside the one-standard-error band, not the maximum.
tree_band = one_se_band(tree_flex)
depth_label = tree_band["complexity"].idxmin()
BEST_DEPTH = None if depth_label == "none" else int(depth_label)
print(f"\n{len(tree_band)} depths within one standard error of the best"
      f" -> simplest is depth {depth_label}")
```

Expected output: an eight-row table carrying a complexity rank, mean training AUC, mean cross-validated AUC, the fold-to-fold standard deviation, and the training-minus-cross-validated gap; then how many depths fell inside the one-standard-error band and which shallowest member was selected.

Input: the training customers only. Transformation: eight tree fits, each cross-validated on the shared stratified folds, then one selection rule applied to the table. Output: the flexibility curve of Section 8.8 in classification units, and a depth chosen by a rule rather than by an eye. Read the gap column first: it is the overfitting signature, and the unlimited tree should show the widest gap in the table — training AUC near 1.0, cross-validated AUC well below the best constrained depth. That is the demonstration the draft of this lab performed against the test set, and it is strictly better performed here, because the test set is not a diagnostic instrument and a demonstration that spends it has bought a lesson with the deliverable's only honest grade.

The same rule now chooses k, on the same folds and with the same band. One detail flips: the complexity column carries a negative sign for k-NN, because the two dials run in opposite directions. A deeper tree is more flexible; a *larger* neighborhood is less flexible, since it averages over more customers and drifts toward the base rate. The simplest candidate inside the band is therefore the shallowest tree and the largest k.

Code 9.8. Select k inside the same folds

```
rows = []
for k in [5, 15, 25, 50, 100, 200]:
    knn = make_pipeline(StandardScaler(),
                        KNeighborsClassifier(n_neighbors=k))
    s = cross_validate(knn, X_tr, y_tr, cv=CV,
                       scoring={"auc": auc_scorer},
                       return_train_score=True)
    rows.append({"k": k, "complexity": -k,   # larger k = simpler model
                 "train_auc": s["train_auc"].mean(),
                 "cv_auc": s["test_auc"].mean(),
                 "cv_sd": s["test_auc"].std(),
                 "gap": s["train_auc"].mean() - s["test_auc"].mean()})
knn_flex = pd.DataFrame(rows).set_index("k")
print("\n" + knn_flex.round(3).to_string())

knn_band = one_se_band(knn_flex)
BEST_K = int(knn_band["complexity"].idxmin())
print(f"\n{len(knn_band)} values of k within one standard error of the"
      f" best -> simplest (largest k) is k = {BEST_K}")
```

Expected output: a six-row table with the same five columns, then the size of the one-standard-error band and the selected value of k.

Input and transformation as above, with six neighborhood sizes in place of eight depths. Output: the second of the two settings the leaderboard needs, chosen by the same declared rule. Both cells share one fold generator and one band function, which is what makes the two selections comparable and what keeps the same-split clause of Section 8.9 true across them.

VERIFICATION CHECK

Before you run: predict three shapes. Training AUC as depth increases — state the direction and whether it is guaranteed or merely expected. Cross-validated AUC as depth increases — predict where the turn falls, and write down what you will conclude if no turn appears within the tested range. And k-NN at k = 5 against k = 200 — say which end you expect to overfit, which end you expect to collapse toward the base rate, and why the second is the majority-class baseline rediscovered as a limiting case. Then predict how many candidates will fall inside the one-standard-error band.

After you run: grade every prediction, then check the rule's work by hand. Confirm that the selected depth is the shallowest inside the band rather than the highest scoring, and say in one sentence what the rule bought: a model whose behavior is easier to explain, monitor, and retrain, at a cost in cross-validated AUC that the band declares immaterial. Name the two sides of each curve in Section 8.8's vocabulary and state what the unlimited tree and k = 5 are actually memorizing.

Investigate if: the band contains every candidate. That means the fold-to-fold noise swamps every difference between settings, which is a legitimate finding on a small training population — report it, take the simplest candidate, and say plainly that the flexibility dial did not matter here.

9.12.5 Lab 9.2, Part B: The Training-Only Leaderboard

The leaderboard was fixed in the frame cell: cross-validated AUC on shared stratified folds, five candidates, one fold generator, a selection rule written before the race. One change from the draft of this lab is worth stating in advance, because it decides what the leaderboard can and cannot conclude. The leaderboard grades **ranking only**. Expected net value does not appear on it, because a dollar figure computed from raw model scores is denominated in probabilities nobody has checked, and Section 9.8's own rule forbids that. The economics arrive in Part C, once the finalist has been calibrated inside the training data, and they arrive as a gate the selected model must pass rather than as a column models compete on.

The two baselines are candidates rather than commentary — fitted, scored, and ranked by the identical call, because a baseline evaluated by a different route is not evidence about the same question — and both now come from the library rather than from hand-written estimator classes. The constant-score floor is scikit-learn's `DummyClassifier` predicting the training prior for everyone; its AUC is 0.5 by construction, and printing it is a check that the scoring plumbing works. The continuous-recency baseline is a logistic regression on recency alone. That choice is deliberate and repairs a real defect in the obvious alternative: a baseline that scores customers by rescaling recency between the smallest and largest values *in the data being scored* would define its own scale from the evaluation population, so a customer's score would depend on who happened to be scored alongside her, and every fold would use a different ruler. A recency-only logistic model fits its parameters on the training data like any other model, produces genuine probabilities that can be checked for calibration, and — if it turns out to be the best thing available — can honestly carry the economic threshold.

The ninety-day rule itself is not on this leaderboard, and the omission is deliberate. A binary 0/1 flag has only one nontrivial operating point, so its AUC is not comparable to the AUC of a continuous score in the way a shared column implies; the rule is evaluated as a fixed policy in Part C and Part E, on the confusion matrix, precision, recall, treated count, and expected net value that are the terms it actually competes in.

Code 9.9. Register the candidates and declare the selection rule

```
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression

COST, BENEFIT = 12.0, 40.0
THRESHOLD = COST / BENEFIT          # 0.30, derived in Section 9.9
RECENCY_ONLY = ["recency_days"]

constant_floor = DummyClassifier(strategy="prior")
recency_logit  = make_pipeline(StandardScaler(),
                               LogisticRegression(max_iter=1000))
logistic = make_pipeline(StandardScaler(),
                         LogisticRegression(max_iter=1000))
tree = DecisionTreeClassifier(max_depth=BEST_DEPTH, random_state=42)
knn  = make_pipeline(StandardScaler(),
                     KNeighborsClassifier(n_neighbors=BEST_K))

# Each candidate is paired with the columns it is allowed to see -- which,
```

```
# read the other way, is a written record of what it is forbidden to see.
BASELINE = "recency-only logistic (baseline)"
CANDIDATES = {
    "constant score (floor)":  (constant_floor, FEATURES),
    BASELINE:                  (recency_logit,  RECENCY_ONLY),
    "logistic regression":     (logistic,       FEATURES),
    f"tree (depth {BEST_DEPTH})": (tree,        FEATURES),
    f"k-NN (k = {BEST_K})":    (knn,            FEATURES),
}

# THE SELECTION RULE, declared here -- before the race is scored -- and
# executed unedited in Code 9.11 and Code 9.13.
MIN_AUC_MARGIN = 0.02      # AUC a model must add over the baseline
MIN_FOLDS_WON  = 4         # of five, on paired per-fold differences
SIMPLICITY_ORDER = ["logistic regression", f"tree (depth {BEST_DEPTH})",
                    f"k-NN (k = {BEST_K})"]
MAX_CALIBRATION_ERROR = 0.05   # near the threshold, on out-of-fold scores

for name, (_, cols) in CANDIDATES.items():
    assert set(cols) <= set(train_df.columns), name
    for forbidden in ("churned", "outcome_orders", "spend_next6m"):
        assert forbidden not in cols, f"{name} was handed {forbidden}"
print("candidates registered:", len(CANDIDATES),
      "| economic threshold:", THRESHOLD)
```

Expected output: the candidate count and the derived threshold of 0.3, after fifteen structural assertions confirm that every candidate's column list exists and that none of them contains the label or its ingredients.

Input: the two settings chosen in Code 9.7. Transformation: two library baselines and three pipelines, paired with their permitted columns. Output: a registry, and a selection rule written before any candidate is scored. That last object is the cell's real content: Section 8.9's leaderboard discipline is enforceable only if the rule predates the results, and a rule typed after the table has been read is a preference wearing a rule's clothes. Note that the rule now has four clauses rather than three — an AUC margin, a folds-won requirement, a simplicity order, and a calibration tolerance — because Section 9.8 made calibration a condition on the economics rather than a nicety.

Code 9.10. Build the training-only ranking leaderboard

```
rows, fold_auc = [], {}
for name, (est, cols) in CANDIDATES.items():
    s = cross_validate(est, train_df[cols], y_tr, cv=CV,
                       scoring={"auc": auc_scorer})
    fold_auc[name] = s["test_auc"]
    rows.append({"candidate": name,
                 "features": len(cols),
                 "cv_auc": s["test_auc"].mean(),
                 "auc_sd": s["test_auc"].std()})

leaderboard = pd.DataFrame(rows).set_index("candidate")
leaderboard["auc_margin_vs_baseline"] = (
    leaderboard["cv_auc"] - leaderboard.loc[BASELINE, "cv_auc"])
print(f"training base rate: {y_tr.mean():.3f}   "
      f"majority-class accuracy: {max(y_tr.mean(), 1 - y_tr.mean()):.3f}")
print(leaderboard.round(3).to_string())
assert abs(leaderboard.loc["constant score (floor)",
                           "cv_auc"] - 0.5) < 0.01
print("\nNo dollar figure appears on this table by design: expected value"
      "\nis computed in Code 9.12, from calibrated probabilities.")
```

Expected output: the base rate and majority-class floor in a header line, then a five-row table of feature count, cross-validated AUC, its fold-to-fold spread, and each candidate's AUC margin over the recency-only baseline; one assertion confirms the constant scorer lands at 0.5, and a closing note records why no net-value column is present.

Input: the training customers only. Transformation: five-fold stratified cross-validation of every candidate on one shared fold generator, each scored on the columns its registry entry permits. Output: the ranking evidence, and the only evidence the analyst is permitted to use when choosing a model family. Read the standard-deviation column as seriously as the mean, exactly as Chapter 8 instructed: a margin smaller than the fold-to-fold spread of the metric it is measured on is not yet a finding.

VERIFICATION CHECK

Before you run: write four predictions. Where will the recency-only baseline's AUC land, and why is a value well above 0.5 the *expected* result rather than a disappointment? Which of the three multivariable models wins, and by how much over the baseline? Is that margin larger than the fold-to-fold spread? And what would an implausibly high AUC look like on this base, given the volatility the label's construction implies?

After you run: grade every prediction, then audit your own table against Section 8.9's three clauses. Same split: confirm that one fold generator produced every row. Same metric: one column, declared in advance, with no dollar figure available to switch to. Same baseline: state each model's margin over the recency-only baseline and say whether it exceeds the spread. Then write the sentence this leaderboard licenses and the sentence it does not: it licenses a claim about which family orders customers best, and it licenses nothing at all about money.

Investigate if: any candidate's cross-validated AUC lands near or above 0.9. Against a base whose churn label summarizes six months of individual human behavior, that is Section 8.4's too-good signature, and the feature trace comes before the celebration — starting with the recency range check of Section 8.12, because a recency below 181 days at a December 31 snapshot is arithmetically impossible.

9.12.6 Lab 9.2, Part C: The Finalist, Calibrated, and the Economic Gate

Part C is where the chapter's own rule about calibration becomes a step in the pipeline rather than a paragraph of advice. The sequence has six moves and the order is the argument: choose the family from training-only ranking evidence; name the provisional finalist; calibrate *that* finalist entirely inside the training population; obtain out-of-fold calibrated probabilities; evaluate reliability and expected net value on those probabilities; and only then decide whether the program is fundable. Nothing in that sequence touches the test set, and every dollar figure it produces comes from probabilities that were checked before they were multiplied.

Code 9.11. Apply the selection rule and calibrate the finalist

```
from sklearn.base import clone
from sklearn.calibration import CalibratedClassifierCV

base = fold_auc[BASELINE]
rows = []
for name in SIMPLICITY_ORDER:
    diff = fold_auc[name] - base
    rows.append({"candidate": name,
                 "mean paired AUC margin": diff.mean(),
                 "sd of paired margins": diff.std(),
                 "folds won": int((diff > 0).sum()),
                 "clears the rule": bool(diff.mean() >= MIN_AUC_MARGIN
                                         and (diff > 0).sum()
```

```
                                            >= MIN_FOLDS_WON)})
paired = pd.DataFrame(rows).set_index("candidate")
print(paired.round(4).to_string())

qualified = [n for n in SIMPLICITY_ORDER
             if paired.loc[n, "clears the rule"]]
FALLBACK = not qualified
FINALIST = qualified[0] if qualified else BASELINE
print("\nqualifying, simplest first:", qualified or "none")
print("FINALIST:", FINALIST)
if FALLBACK:
    print("No multivariable model cleared the declared rule. The finalist"
          "\nis the recency-only baseline -- itself a fitted probability"
          "\nmodel, so it can be calibrated and thresholded honestly, but"
          "\nthe deliverable's recommendation changes: see Code 9.12.")

# Calibration is fitted INSIDE the training population, on predictions the
# underlying classifier did not make about its own fitting data
# (scikit-learn developers, 2026a). This object -- estimator PLUS
# calibrator -- is what gets frozen and what gets graded.
finalist_estimator, FINALIST_COLS = CANDIDATES[FINALIST]
final_model = CalibratedClassifierCV(clone(finalist_estimator),
                                     cv=CV, method="isotonic")
print("\nfinalist columns:", len(FINALIST_COLS),
      "| calibrator: isotonic, fitted inside the training folds")
```

Expected output: a three-row paired-margin table with the folds each candidate won and whether it clears the declared rule; the qualifying list in simplicity order; the named finalist; the fallback notice if no multivariable model qualified; and a confirmation of the finalist's feature count and calibration method.

Input: the per-fold AUCs stored by Code 9.10. Transformation: fold-by-fold subtraction against the baseline, one rule executed unedited, and one calibration wrapper. Output: a finalist arrived at by a rule rather than a preference, and — crucially — a finalist that is now a *pair*, estimator plus calibrator, which is the object every later cell scores, freezes, and deploys. The comparison is paired for the reason Section 8.9 gave: folds differ in difficulty, a fold holding an unusual mix of lumpy occasion shoppers is hard for every candidate at once, and subtracting the baseline fold by fold cancels the shared difficulty that each candidate's own standard deviation contains.

Code 9.12. Build the out-of-fold reliability exhibit

```
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_predict

def net_per_1000(y_true, p, threshold=THRESHOLD):
    """Expected net value per 1,000 evaluated customers, so that folds
    and populations of different sizes stay comparable."""
    y_true = np.asarray(y_true)
    treat = (np.asarray(p) >= threshold).astype(int)
    tp = int(((treat == 1) & (y_true == 1)).sum())
    treated = int(treat.sum())
    return 1000.0 * (tp * BENEFIT - treated * COST) / len(y_true)

# Out-of-fold probabilities from the CALIBRATED pair. Because final_model
# calibrates internally, this is a nested loop: the calibrator of each
# outer fold never sees that fold's held-out customers.
X_fin = train_df[FINALIST_COLS]
oof = cross_val_predict(final_model, X_fin, y_tr, cv=CV,
                        method="predict_proba")[:, 1]

edges = np.linspace(0.0, 1.0, 11)
cal = (pd.DataFrame({"p": oof, "y": y_tr.to_numpy()})
```

```
        .assign(bin=lambda d: pd.cut(d["p"], edges, include_lowest=True))
        .groupby("bin", observed=True)
        .agg(customers=("y", "size"),
             mean_predicted=("p", "mean"),
             observed_rate=("y", "mean")))
cal["difference"] = cal["observed_rate"] - cal["mean_predicted"]
print(cal.round(3).to_string())

fig, ax = plt.subplots(figsize=(4.6, 4.6))
ax.plot([0, 1], [0, 1], linestyle="--", linewidth=1,
        label="perfect calibration")
ax.plot(cal["mean_predicted"], cal["observed_rate"], marker="o",
        label="out of fold, training customers")
ax.axvline(THRESHOLD, linewidth=1, linestyle=":",
           label=f"economic threshold {THRESHOLD:.2f}")
ax.set_xlabel("mean predicted churn probability")
ax.set_ylabel("observed churn rate")
ax.set_title("Reliability of the calibrated finalist")
ax.legend(fontsize=8)
plt.tight_layout(); plt.show()
```

Expected output: a ten-row reliability table carrying each bin's customer count, mean predicted probability, observed churn rate, and their difference; then a reliability plot with the diagonal and the economic threshold marked.

Input: the calibrated finalist and the training customers. Transformation: nested cross-validated probability predictions, binned against observed rates. Output: the evidence that entitles Section 9.9's arithmetic to be applied. Three details are deliberate. The probabilities are out of fold and come from the calibrated pair, so the plot describes the object that will actually be deployed rather than an uncalibrated ancestor of it. Because `final_model` calibrates internally, the call is a nested loop: the calibrator inside each outer fold never sees that fold's held-out customers. And the customer count per bin is printed because a bin holding eleven customers cannot support a calibration verdict, and a reliability plot without counts invites confident conclusions from empty bins.

The exhibit is evidence; the next cell is the decision. Its two conditions were declared in Code 9.9, both are computed from training customers only, and both must hold before any money is discussed.

Code 9.13. Clear the economic gate

```
# THE GATE. Two conditions, both declared in Code 9.9, both computed from
# training customers only, both required before any money is discussed.
left  = np.array([b.left for b in cal.index], dtype=float)
right = np.array([b.right for b in cal.index], dtype=float)
near = cal[(left >= THRESHOLD - 0.2) & (right <= THRESHOLD + 0.2)]
worst = (float(near["difference"].abs().max())
         if len(near) else float("nan"))

model_net = net_per_1000(y_tr, oof)
incumbent = (train_df["recency_days"] > 90).astype(float)
incumbent_net = net_per_1000(y_tr, incumbent, threshold=0.5)

print(f"\nworst calibration deviation near the threshold: {worst:.3f}"
      f"  (tolerance {MAX_CALIBRATION_ERROR})")
print(f"cross-validated net per 1,000, calibrated:  ${model_net:,.0f}")
print(f"cross-validated net per 1,000, incumbent:   "
      f"${incumbent_net:,.0f}")

GATE_PASSED = (worst <= MAX_CALIBRATION_ERROR
               and model_net > max(0.0, incumbent_net))
print("\nECONOMIC GATE:", "PASSED" if GATE_PASSED else "NOT PASSED")
if not GATE_PASSED:
    print("The scored program is not yet fundable on training evidence."
```

```
          "\nHonest responses, in order: recalibrate or refit inside the"
          "\ntraining folds; retain the ninety-day incumbent policy;"
          "\nrecommend rank-based use with no probability threshold; or"
          "\nreturn to framing and features. Do NOT open the test set"
          "\nlooking for a better number.")
```

Expected output: the worst calibration deviation near the threshold against its declared tolerance, the calibrated model's and the incumbent's cross-validated net value per thousand customers, and the gate's verdict with its consequences.

Input: the out-of-fold calibrated probabilities and the training customers. Transformation: two declared tests — a calibration tolerance near the threshold, and an expected-net-value comparison against zero and against the incumbent. Output: the first place in the lab where a dollar figure legitimately appears, and the decision about whether the scored program is fundable at all. Three details are deliberate. Both tolerances were declared in Code 9.9 rather than chosen by looking at this output, which is what makes this a gate rather than a reaction. The incumbent's net is computed on the same training customers as a fixed policy that fits nothing, so comparing against it here costs nothing and leaks nothing. And the failure branch prints its own honest responses, none of which is "open the test set and look for a better number" — because a gate that can be appealed to the sealed data is not a gate.

VERIFICATION CHECK

Before you run: predict the shape of the curve — points above the diagonal mean the model under-predicts churn, below means it over-predicts — and predict whether the calibrated finalist will beat the ninety-day rule on cross-validated net value. Then write the sentence you will owe the deliverable either way: the analytically derived threshold of 0.30 is valid only when the probabilities are sufficiently calibrated for the deployment population.

After you run: read the two bins that straddle 0.30 first, because those are the bins the decision actually depends on; a model beautifully calibrated at 0.05 and off by fifteen points at 0.30 is miscalibrated where it matters. Then state, in one sentence, what would happen to the treated-list size and the expected net if the scores near the cut were systematically ten points too high — and note that AUC would not have moved at all. Finally, record the gate's verdict in the notebook *before* running anything else, because a gate whose result is read after the test set is open is not a gate.

Investigate if: the gate fails. That is a legitimate and reportable outcome, not a failed lab, and the honest responses are printed by the cell itself. Investigate also if the top bins are empty or nearly so: that is ordinary on a base whose churn probabilities concentrate below 0.6, and it means the verdict rests on the lower bins — report the sparsity rather than reading a confident line through three customers.

9.12.7 Lab 9.2, Part D: The Sealed Test Set, Opened Once

Everything so far happened inside the training customers: the family was chosen on ranking evidence, the finalist was calibrated, and the economics were gated. Part D spends the test set — once, on the frozen estimator-plus-calibrator pair.

Code 9.14. Freeze the model and open the sealed test set once

```
# Refit the pair on ALL training customers. The cross-validated estimates
# came from models fitted on four folds at a time; this is the object that
# would actually be deployed.
final_model.fit(X_fin, y_tr)
```

```
y_te = test_df["churned"]
p_te = final_model.predict_proba(test_df[FINALIST_COLS])[:, 1]

baseline_model = CalibratedClassifierCV(
    clone(CANDIDATES[BASELINE][0]), cv=CV, method="isotonic")
baseline_model.fit(train_df[RECENCY_ONLY], y_tr)
p_base = baseline_model.predict_proba(test_df[RECENCY_ONLY])[:, 1]

tag = "  (fallback -- no model cleared the rule)" if FALLBACK else ""
print(f"finalist: {FINALIST}{tag}")
print(f"training-evidence gate: "
      f"{'PASSED' if GATE_PASSED else 'NOT PASSED'}")
print(f"\ntest base rate:               {y_te.mean():.3f}")
print(f"test majority-class accuracy: "
      f"{max(y_te.mean(), 1 - y_te.mean()):.3f}")
print(f"test AUC, finalist:           {roc_auc_score(y_te, p_te):.3f}")
print(f"test AUC, recency baseline:   {roc_auc_score(y_te, p_base):.3f}")
print("\nTEST SET SPENT. Any further development returns to the folds.")
```

Expected output: the finalist's name with its fallback and gate status, then four test-set figures — base rate, majority-class accuracy, and two AUCs — followed by the spending notice.

Input: the training customers, the frozen pair, and the sealed test customers. Transformation: one refit and two scoring passes. Output: the deliverable's central grade. Two details enforce disciplines the chapter argued for. The finalist and the gate verdict are printed *above* the test figures, so the notebook itself records that the decision preceded the grade rather than following it. And the baseline is calibrated too, so the two AUC figures compare like with like and the baseline remains usable as an honest fallback rather than as a scoring convenience.

VERIFICATION CHECK

Before you run: write four predictions. How close will the test AUC sit to the finalist's cross-validated AUC, and in which direction do you expect them to differ? What would an alarming result look like in each direction — too poor, and too good? Will the calibrated finalist beat the calibrated recency baseline on the test customers by roughly the margin the folds suggested? And what will you do if it does not?

After you run: place the result on Table 8.4's signature grid, then write the two sentences that survive into the Section 9.13 deliverable: the ranking sentence, which states the finalist's test AUC beside the baseline's, and the floor sentence, which states the test base rate and the majority-class accuracy every later accuracy figure must be read against. Then honor the protocol: the test set is now spent, and any new idea returns to the training folds.

Investigate if: the test AUC sits far above the cross-validated figure. A gap larger than the fold-to-fold spread usually means a column entered the feature list after the leaderboard was frozen, or the feature table was rebuilt at a later snapshot. Both are found by rerunning Code 9.3 from the top and confirming that the row counts and the recency minimum are unchanged.

9.12.8 Lab 9.2, Part E: Five Policies, One Table

Part E is the miniature's strongest lesson at full scale. The threshold was derived before any confusion matrix was printed, in a markdown cell the CFO could read — cost per treatment $12; expected benefit per treated true churner $40 (save rate 0.25 × retained margin $160, both flagged as finance estimates); therefore treat above p = 12 ÷ 40 = 0.30 — and the reliability evidence of Code 9.12 is what entitles that cut to be applied to these scores. One reading rule attaches before the table is printed: if the economic gate of Code 9.12 did not pass, this table is evidence *against*

funding the scored program rather than a menu to choose from, and the deliverable reports it that way. Then, and only then, the matrices.

Code 9.15. Compare five policies on the sealed test customers

```
scored = test_df.assign(p=p_te)
scored["lapsed_90d"] = (scored["recency_days"] > 90).astype(int)

def price_full(name, treat, truth=scored["churned"]):
    treat = pd.Series(np.asarray(treat), index=truth.index).astype(int)
    tp = int(((treat == 1) & (truth == 1)).sum())
    fp = int(((treat == 1) & (truth == 0)).sum())
    fn = int(((treat == 0) & (truth == 1)).sum())
    tn = int(((treat == 0) & (truth == 0)).sum())
    treated = tp + fp
    return {"policy": name, "TP": tp, "FP": fp, "FN": fn, "TN": tn,
            "treated": treated,
            "accuracy": (tp + tn) / len(truth),
            "precision": tp / treated if treated else float("nan"),
            "recall": tp / (tp + fn) if (tp + fn) else float("nan"),
            "net": tp * BENEFIT - treated * COST}

policy_table = pd.DataFrame([
    price_full("treat nobody", 0),
    price_full("ninety-day rule (incumbent)", scored["lapsed_90d"]),
    price_full("model at 0.50 (software default)",
               (scored["p"] >= 0.50).astype(int)),
    price_full(f"model at {THRESHOLD:.2f} (derived)",
               (scored["p"] >= THRESHOLD).astype(int)),
    price_full("treat everyone (blanket)", 1),
]).set_index("policy")

_br = scored["churned"].mean()
print(f"test base rate {_br:.3f} | "
      f"majority-class accuracy {max(_br, 1 - _br):.3f}")
print(policy_table.round(2).to_string())

# Sensitivity: the threshold inherits its inputs' uncertainty (9.9).
for benefit in (30.0, 40.0, 60.0):
    t = COST / benefit
    treat = (scored["p"] >= t).astype(int)
    tp = int(((treat == 1) & (scored["churned"] == 1)).sum())
    print(f"benefit ${benefit:>5.0f} -> threshold {t:.2f}: "
          f"treated {int(treat.sum()):>5d}   "
          f"net ${tp * benefit - int(treat.sum()) * COST:,.0f}")
```

Expected output: a header line carrying the test base rate and majority-class accuracy, a five-row policy table with the four confusion counts and five metrics per row, and three sensitivity lines showing how the threshold, the treated-list size, and the expected net move as the benefit estimate varies.

Input: the sealed test customers and their calibrated scores. Transformation: five treatment vectors run through one pricing function, then the same arithmetic at three benefit estimates. Output: the exhibit the meeting decides on, and the sensitivity the meeting is owed. The precision column carries NaN in the do-nothing row rather than raising a division error, which is the same safe-denominator handling Code 9.2 established and the reason the function is written once rather than five times.

VERIFICATION CHECK

Before you run: predict the direction of every movement from the 0.50 row to the 0.30 row. The treated list must grow, recall must rise, precision must fall, and — on data whose economics match the miniature's shape — the expected net must improve, because the program's dear error is the false negative and the lower cut buys fewer of them. Predict also whether the ninety-day rule's net is positive, and whether the blanket's net is positive; the second is arithmetic you can do from the base rate alone, since treating everyone earns 40 × (base rate) and costs 12 per customer.

After you run: re-add the four cells of every row by hand and reconcile each to the test-set count — a matrix that does not sum is a pasted matrix, and pasted matrices are how wrong numbers reach decks. Then write the executive sentence the chapter has been building toward: at the 0.30 threshold, precision below one half means most treated customers will not have churned *by design*, because the program deliberately overspends on the cheap error to avoid the dear one. Finally, read the sensitivity block: if the verdict flips within finance's plausible benefit range, the binding uncertainty is the economics rather than the model.

Investigate if: the derived-threshold row loses money on the test customers after passing the training gate. That is the gap between a cross-validated estimate and a single held-out sample, and the honest report gives both figures with their populations named rather than quoting whichever is friendlier.

9.12.9 Lab 9.2, Part F: The Ranked-Targeting Exhibits

Part F builds the three charts the chapter owns and the decile table beneath them. These are analytic figures rather than presentation graphics; Chapter 12 supplies the visualization theory, and this lab supplies the arithmetic the theory will later dress.

Code 9.16. Construct the decile table

```
from sklearn.metrics import roc_curve

scored["decile"] = pd.qcut(
    scored["p"].rank(method="first", ascending=False),
    10, labels=range(1, 11)).astype(int)

base_rate = scored["churned"].mean()
dec = (scored.groupby("decile")["churned"]
       .agg(customers="count", churners="sum", rate="mean"))
dec["lift"] = (dec["rate"] / base_rate).round(2)
dec["cum_capture"] = (dec["churners"].cumsum()
                      / dec["churners"].sum()).round(3)
dec["cum_file"] = (dec["customers"].cumsum()
                   / dec["customers"].sum()).round(3)
print(dec.to_string())
assert np.isclose(dec["cum_capture"].iloc[-1], 1.0)

# The incumbent as one operating point on the same axes.
inc = scored["lapsed_90d"] == 1
inc_file    = inc.mean()
inc_capture = scored.loc[inc, "churned"].sum() / scored["churned"].sum()
```

Expected output: a decile table with customer counts, churner counts, churn rate, lift, cumulative capture, and cumulative file share, and one assertion that capture reaches 1.0.

Input: the scored test customers. Transformation: a rank-based decile assignment and three cumulative computations. Output: the arithmetic beneath all three charts. The decile assignment ranks before cutting —

`rank(method="first")` breaks ties deterministically — so the ten groups are equal in size regardless of how many customers share a score, which is what makes lift comparable across deciles. The incumbent's file share and capture are computed here too, because the next cell needs them as a single point on two different sets of axes.

Code 9.17. Plot the ROC, lift, and gains exhibits

```
fig, axes = plt.subplots(1, 3, figsize=(13.5, 4.2))

fpr_m, tpr_m, _ = roc_curve(scored["churned"], scored["p"])
fpr_r, tpr_r, _ = roc_curve(scored["churned"], p_base)
axes[0].plot(fpr_m, tpr_m, label="finalist")
axes[0].plot(fpr_r, tpr_r, linestyle="-.", label="recency-only baseline")
axes[0].plot([0, 1], [0, 1], linestyle="--", linewidth=1, label="chance")
inc_fpr = ((inc) & (scored["churned"] == 0)).sum() / (
    scored["churned"] == 0).sum()
axes[0].scatter([inc_fpr], [inc_capture], marker="D", zorder=5,
                label="ninety-day rule")
axes[0].set_xlabel("false positive rate"); axes[0].set_ylabel("recall")
axes[0].set_title("ROC"); axes[0].legend(fontsize=8)

axes[1].bar(dec.index, dec["lift"])
axes[1].axhline(1.0, linestyle="--", linewidth=1)
axes[1].set_xlabel("decile (1 = highest)"); axes[1].set_ylabel("lift")
axes[1].set_title("Lift by decile, reference line at 1.0")

axes[2].plot([0] + list(dec["cum_file"]), [0] + list(dec["cum_capture"]),
             marker="o", label="model")
axes[2].plot([0, 1], [0, 1], linestyle="--", linewidth=1,
             label="random targeting")
axes[2].scatter([inc_file], [inc_capture], marker="D", zorder=5,
                label="ninety-day rule")
treated_share = (scored["p"] >= THRESHOLD).mean()
axes[2].axvline(treated_share, linestyle=":", linewidth=1,
                label=f"{THRESHOLD:.2f} cut treats "
                      f"{treated_share:.0%} of the file")
axes[2].set_xlabel("share of file treated")
axes[2].set_ylabel("share of churners captured")
axes[2].set_title("Cumulative gains"); axes[2].legend(fontsize=8)

plt.tight_layout(); plt.show()
print(f"ninety-day rule: treats {inc_file:.1%} of the file, "
      f"captures {inc_capture:.1%} of churners")
```

Expected output: a three-panel figure carrying the ROC curves with the incumbent marked, lift by decile against its 1.0 reference line, and cumulative gains against the random-targeting diagonal with the economic threshold's treated share marked; then the incumbent's file share and capture printed for the deliverable.

Input: the decile table and the scored file. Transformation: three plots on shared data. Output: the exhibits the budget conversation runs on. The panels answer three different questions and should be read in that order — whether the ordering is any good at all, how concentrated the churners are at the top of it, and how much of the problem a given depth of file reaches. The ninety-day rule appears on the first and third panels as a single marked point rather than a curve, which is the visual statement of why it does not belong in an AUC column: a fixed policy has one operating point, and a scored file has a curve of them.

VERIFICATION CHECK

Before you run: predict the table's shape. Decile 1's churn rate should run well above the base rate, so its lift should sit meaningfully above 2 for a model near this data's ceiling; the bottom deciles should run far below the base rate, which is the half of the exhibit executives forget, because the model is also identifying customers who *do not need* the program. Predict where the gains curve crosses fifty percent capture, and predict whether the ninety-day rule's diamond will sit above or below the model's curve at the same file share.

After you run: read lift with the right expectation. Lift should generally be strongest in the early deciles and weaken as the list deepens; perfect monotonic decline is not guaranteed in a finite test sample, and a non-monotone middle is a finding about where the ordering blurs rather than an error. What would be a failure is a top decile whose lift sits near 1.0. Then write the three sentences the meeting will quote. The capture sentence: the top two deciles contain X percent of all churners — reconcile it against the cum_capture column by hand. The comparison sentence: the ninety-day rule treats Y percent of the file and captures Z percent, placed on the same axes. The budget sentence: at $12 per treatment, funding through decile 2 costs $A and reaches B expected saves worth $C, and the marginal decile stops paying when its incremental capture times $40 falls below its treatment cost.

Investigate if: the economic threshold's vertical line falls far to the right of the deciles the budget can fund. That is the capacity-versus-threshold tension of Section 9.9 arriving in visual form, and the deliverable owes the meeting the priced gap: what the unfunded treatments are expected to return, which is the argument for more budget stated in the only currency budget arguments accept.

9.12.10 Lab 9.2, Part G: The Fairness Appendix and the AI Round Trip

Part G closes the lab with the two audits the deliverable cannot ship without. The first is the fairness appendix proper — not a comparison of average scores, which is what a first draft usually produces, but per-group error rates at the threshold the program will actually use. Section 9.15 argues, and this exhibit shows, that a model can be honest on average and systematically wrong about an identifiable group.

Code 9.18. Audit group-level errors and score behavior

```
sc = (scored.join(customers.set_index("customer_id")
                  [["primary_store_id", "home_metro"]])
      .merge(stores[["store_id", "opening_date"]],
             left_on="primary_store_id", right_on="store_id",
             how="left"))

# Newest-store cohort: primary store opened in the 12 months before the
# snapshot. Online-only customers have no store: False, not missing.
sc["new_store"] = sc["opening_date"].ge(
    pd.Timestamp("2025-01-01")).fillna(False)

# A declared lifecycle grouping built only from snapshot-safe columns.
sc["lifecycle"] = np.select(
    [sc["frequency"].eq(1), sc["frequency"].between(2, 3)],
    ["single-purchase", "developing"], default="established")

def group_report(df, by, threshold=THRESHOLD):
    """Treatment rate and both error rates by group, at the threshold the
    program will use. Mean tenure is reported so that any residual
    imbalance in the comparison stays visible. Denominators are checked
    before they divide."""
    out = []
```

```
    for key, g in df.groupby(by, dropna=False):
        treat = (g["p"] >= threshold).astype(int)
        y = g["churned"]
        tp = int(((treat == 1) & (y == 1)).sum())
        fp = int(((treat == 1) & (y == 0)).sum())
        fn = int(((treat == 0) & (y == 1)).sum())
        tn = int(((treat == 0) & (y == 0)).sum())
        out.append({
            by: key, "customers": len(g),
            "mean_tenure": g["tenure_days"].mean(),
            "base_rate": y.mean(), "mean_score": g["p"].mean(),
            "treat_rate": float(treat.mean()),
            "precision": tp / (tp + fp) if (tp + fp) else np.nan,
            "recall":    tp / (tp + fn) if (tp + fn) else np.nan,
            "fpr":       fp / (fp + tn) if (fp + tn) else np.nan,
            "fnr":       fn / (fn + tp) if (fn + tp) else np.nan})
    return pd.DataFrame(out).set_index(by).round(3)

for dimension in ["new_store", "home_metro", "lifecycle"]:
    print(f"\n--- by {dimension} ---")
    print(group_report(sc, dimension).to_string())

# The artifact test: repeat the new-store comparison inside a narrow
# tenure band. This RESTRICTS the comparison; it does not match it, so
# the mean_tenure column is what tells you how much imbalance survives.
young = sc[sc["tenure_days"] < 365]
print(f"\n--- by new_store, tenure under one year "
      f"(n = {len(young):,}) ---")
print(group_report(young, "new_store").to_string())
```

Expected output: three group tables — by new-store status, by metro, and by lifecycle group — each carrying group size, mean tenure, base rate, mean score, treatment rate, precision, recall, false-positive rate, and false-negative rate; then the same new-store comparison recomputed within customers of under one year's tenure.

Input: the scored test customers joined to the customer and store dimensions. Transformation: three declared groupings and one tenure restriction, each run through identical error arithmetic. Output: the deliverable's fairness appendix. Three construction details carry the section's argument. The new-store flag treats an unmatched store — an online-only customer, whose primary_store_id is the reserved ONLINE value — as False rather than missing, and the choice is declared rather than silent. The lifecycle grouping is built from snapshot-safe behavioral columns rather than from Chapter 6's fitted segments, so the appendix profiles the treated population without importing a learned assignment into a pipeline that never trained on one. And the mean-tenure column exists because the final comparison is a *restriction*, not a match: confining both groups to under a year does not equalize their tenure distributions, and a restricted comparison that still shows one group averaging four months against the other's ten has not removed the artifact it was meant to remove. Report the residual, and say what a genuine match — finer bands, or a matched-pairs design — would take.

VERIFICATION CHECK

Before you run: predict the new-store rows using nothing but the store roster and the eligibility rule. Customers whose primary store opened recently are, by construction, recent customers — short tenure, low frequency, thin history — and every one of those features raises the model's churn score by design, so predict a higher mean score, a higher treatment rate, and a higher false-positive rate for the newest cohort. Then predict what the tenure-restricted table will do to the gap, and predict how much tenure imbalance will survive inside the band.

After you run: state the finding in Section 7.2's verbs. The model has not discovered that new-market customers are disloyal; it has discovered that they are new. Read the two mean-tenure figures inside the restricted table before reading anything else: if they still differ materially, the restriction has removed only part of the artifact and the remaining gap cannot be attributed cleanly. Then read the metro and lifecycle tables for any group whose false-positive or false-negative rate departs materially from the aggregate, and write one sentence per departure naming what the program would do differently and what it would cost.

Investigate if: any group's row carries fewer than about fifty customers. Error rates computed on small groups swing wildly, and a fairness appendix that reports a 0.00 false-negative rate on a group of nine has produced noise with a decimal point. Report the group size beside every rate, and say so.

One framing sentence belongs in the deliverable beside this table, and it is a limit rather than a caveat. What Code 9.18 produces is a *minimum diagnostic screen*: evidence that the program's errors were examined across identifiable groups before launch, and a record of what was found. It is not a fairness certification. No single error-parity comparison establishes that a treatment is fair, beneficial, or legally compliant, the parity criteria themselves can conflict with one another, and a program can pass every column in this table and still be a bad idea for the people in it. The screen's honest claim is narrow and worth making: nobody launched without looking.

The second audit is the round trip. Hand an assistant the raw tables and the deliberately loose prompt "build me the best churn model you can — maximize accuracy," and grade the response with Table 8.5's five points followed by Table 9.5's four, in writing: reconstruct the churn definition the assistant invented and compare it to Section 9.3's, naming which levers of Table 9.1 it moved; trace its features across the snapshot, running the recency range check; find the threshold it silently applied; compute the base rate and majority-class baseline it did not report; determine whether any setting was chosen by consulting a test split; look for a reliability exhibit and note whether any dollar figure was computed without one; and re-derive its headline metric from its own confusion matrix. The designed likelihood, engineered by the loose prompt: a label built at the analysis date (S3 and Table 8.2 fail together), a 0.5 threshold nobody chose, a depth or a k selected against test performance, uncalibrated scores feeding an expected-value table, and an accuracy headline whose margin over the majority-class floor was never computed. Your audit memo, filed per Appendix D, is the lab's final deliverable — and the direct rehearsal of the platform-flag verdict Section 9.13 writes.

9.13 Marketing Interpretation and Managerial Insight

The lab's outputs are matrices, curves, and a derivation; the twenty-eight-day deadline runs on sentences. This section translates, and — per this guide's standing practice — it does so partly by exhibiting the wrong managerial readings and correcting them, because classification mints misreadings faster than any machinery yet: its numbers sound like everyday words (accurate, precise), its scores sound like verdicts, and both will be spoken in the targeting meeting by people whose only error is trusting the vocabulary.

The first wrong reading arrives holding the platform's documentation: "Their churn flag is 89 percent accurate. Our model's accuracy at the threshold you chose is about the same — and theirs is one click. Why did we build anything?" The correction is not that 89 percent is bad; it is that 89 percent is not yet a number anyone can read. At a base rate near thirty percent, seventy percent accuracy is available for free by treating no one, so the claim's real content is a margin over that floor — a margin that cannot be evaluated without the confusion matrix behind it, the threshold that produced it, and the cost structure that prices the two errors it distributes. And the claim is unauditable besides, because 89 percent accurate *at what definition of churn, at what threshold, on whose customers* are questions the panel's documentation does not answer. Then the reframe the meeting actually needs: accuracy was never the currency. The deliverable's currency is capture against cost — the model's top two deciles reach a stated share of the departing customers at a stated program cost, the platform flag's list (once exported and graded on the same test months) reaches its own share at its own cost, and the two can compete on one gains chart the moment the vendor discloses enough to be graded. The platform verdict is Chapter 8's vendor verdict with this chapter's

supplement: the five questions of Table 8.5, plus Table 9.5's four — and the one that ends most such meetings is S3, because a flag that cannot state its threshold's cost assumption is a business policy no one at StyleCraft agreed to.

The second wrong reading is quieter and does more damage per occurrence: "The model says C003914 has a 0.62 churn probability — she's leaving. Why is she still getting the full-price catalog? And C005108 is at 0.08, so we can stop worrying about him." Both sentences collapse a probability into a verdict, and Section 8.13's discipline transfers with the units changed: 0.62 means that among snapshot-similar customers, roughly six in ten went dark — and four in ten did not. The score is not a prophecy about her; it is a rank and a rate, and the program built on it is justified by the *portfolio* — treat enough 0.62s and the arithmetic of Section 9.9 pays reliably — not by any individual call being right. The corrected sentences change the object of confidence exactly as Chapter 8's did: not "she is leaving" but "she belongs to the group most worth treating"; not "he is safe" but "treating his group costs more than it returns." One implication deserves its own sentence in the deliverable, because it defuses the meeting's most predictable ambush: under a 0.30 threshold, the *majority* of treated customers will not have churned even if the model is excellent — precision below one half means most vouchers land on stayers *by design*, because the program deliberately overspends on the cheap error to avoid the dear one. An executive who has not been told this in advance will read the first post-campaign report as proof the model failed; an executive who has been told will read the same report as the cost structure working as priced. Setting that expectation before launch is not spin — it is the threshold decision's second half, and forgetting it has ended more scoring programs than bad models have.

A third misreading is newer to this chapter and specific to its arithmetic: "The model says 0.30 is the cut, so let's use 0.30." That sentence is correct only if the scores near 0.30 mean thirty percent, which is a property the reliability exhibit of Code 9.12 either establishes or refutes. Where it refutes, the threshold is not wrong — the ledger is still 12 against 40 — but the scores are the wrong ruler for it, and the repair belongs in the training data rather than in the meeting. This is the clearest case in the chapter where a number that survives every ranking diagnostic can still price a decision incorrectly, and it is the reason the deliverable carries the calibration exhibit rather than merely mentioning calibration in a footnote.

The deliverable that survives all three misreadings has a fixed anatomy, assembled entirely from lab outputs. Page one is the churn definition per Section 9.3 — the label, its activity basis, its two eligibility conditions, its rejected alternatives, and who each alternative would have added or removed — because every later number is downstream of it. The training-only leaderboard follows, per Section 8.9's fixed rules: three models against the constant-score floor and the recency-only logistic baseline, graded on cross-validated AUC alone, with the base rate printed in the header, the paired fold margins shown, and no dollar figure anywhere on it. Then the selection rule, executed unedited, and the calibration of the selected model inside the training data. Then the threshold derivation as arithmetic — the $12, the $40, the 0.30, the sensitivity at $30 and $60 — stated so the CFO can re-run it, with the out-of-fold reliability exhibit and the economic gate beside it: the calibrated model's cross-validated net value per thousand customers against zero and against the incumbent rule, computed before the test set was opened. A threshold the CFO has re-derived is a threshold the CFO defends thereafter; a threshold applied to unchecked probabilities is a threshold nobody should defend; and a program that cannot clear its own gate on training evidence is a program the deliverable should decline to fund rather than test its way into. Then the single test-set grade, opened once, on the frozen estimator-plus-calibrator pair. Then the five-policy table at the derived threshold, hand-reconciled, with the precision expectation set in plain words. Then the gains chart with the three policies on one page — model deciles, ninety-day rule, blanket — and the budget sentence the stopping rule produces. The incremental caveat, flagged per Section 7.2 and priced per Section 9.9: the save rate is an assumption this design cannot certify, the experiment that would certify it is named, and the deliverable recommends the program launch *with* a randomly held-out control group, because the cheapest moment to buy causal evidence is before the rollout, not after the doubt. The monitoring plan per Section 8.10, with two classification-specific lines: the base rate itself is monitored, because a drifting base rate silently re-prices the threshold even when the ordering holds; and the calibration is re-checked on each matured cohort, because probabilities drift out of true before rankings do. The permitted-use note per Section 8.15. And the fairness appendix from Lab 9.2 Part G — the per-group error rates, the new-store exhibit, and its tenure-restricted comparison with the residual tenure imbalance reported, framed as a minimum diagnostic screen — which Section 9.15 now takes up on its own terms.

9.14 Business Analytics in Practice

This section turns from the fictional case to how churn scoring, retention economics, and classification governance operate in industry — where churn models are among the oldest production analytics in marketing, where the lift chart is a budget instrument, and where the hardest lessons are about what the model made people do. The vignettes below are drawn from published research and from recurring professional patterns rather than from any single named organization.

9.14.1 The Telecom Churn Playbook and Two Different Warnings

The first vignette is the telecom churn playbook, because telecommunications is where churn modeling grew up and where its most instructive surprises were documented. Contractual carriers have modeled churn for three decades: rich behavioral data, a legally crisp label (the cancellation event this chapter's non-contractual setting had to construct by hand), dedicated save desks, and retention offers whose economics are exactly Section 9.9's ledger at industrial scale. The playbook's mature form looks like this chapter's deliverable — scored base, threshold from offer economics, treatment tiers by decile — and it comes with two distinct pieces of scar tissue that students routinely fuse into one and should not.

The first warning is about *targeting*. Ascarza (2018), combining two field experiments with machine-learning targeting rules, found that the customers a churn model ranks riskiest are not necessarily the customers most *responsive* to a retention intervention, so programs aimed by risk rather than by sensitivity to the treatment can misallocate their budgets — and that targeting on estimated response heterogeneity outperformed the standard practice of targeting the highest-risk customers. In this guide's vocabulary: risk is who is leaving; responsiveness is whom the treatment changes; they are different quantities, and this chapter's observational machinery estimates only the first.

The second warning is stronger and comes from a different study, which is why the two should not be cited interchangeably. Ascarza et al. (2016), in a large-scale field experiment in which some customers received proactive pricing-plan recommendations and some did not, found that the intervention *increased* churn: 10 percent of the treated group churned in the three months following the campaign against 6 percent of the control group, with the authors attributing the effect to lowered inertia about switching and to raised salience of past usage. That is not a modeling error at all — the churn model predicted honestly; the *program* assumed that predicting and preventing are the same problem. Only a held-out control inside the campaign can measure the second, which is Chapter 11's machinery. The practice lesson, planted in Section 9.13's deliverable and repeated here because industry paid heavily to learn it: launch scored programs with controls from day one, and treat "intervention is not prediction" as a design constraint rather than a footnote.

9.14.2 The Lift Chart as a Budget Instrument

The second vignette is the lift chart as a budget instrument, because the exhibit students meet as a model-evaluation chart functions in industry as a finance document. At subscription and retail firms with mature retention programs, the annual retention budget is negotiated *off the gains curve*: the analytics team presents cumulative capture by decile, finance attaches the treatment cost and save economics — the same ledger as Section 9.9, at line-item scale — and the budget conversation becomes a stopping-rule conversation: fund deciles while the marginal decile's expected saved margin clears its treatment cost, stop where the curve's slope says to stop. Teams that operate this way report the cultural shift this chapter's Concept box predicted: the argument stops being "is the model good" — a question finance cannot engage — and becomes "where does the curve stop paying," a question finance is better at than the analysts.

The recurring failure mode is equally instructive: programs that set the treated-list size first (a round number, a platform tier limit, last year's list) and never reconcile it against the curve, so that the budget quietly funds deciles the arithmetic abandoned — the capacity-versus-threshold tension of Section 9.9, resolved by inertia instead of by the priced argument. A second failure is subtler and belongs to this chapter's calibration thread: teams that read the gains curve correctly and then compute the program's expected return from raw model scores that nobody checked for calibration, producing a budget denominated in probabilities the model does not actually print. The lesson is the

closing thread's preview: the decile table is where analytics and finance speak the same language, and the analyst who brings the curve — and the reliability exhibit that entitles its dollar figures — to the budget meeting sets the agenda.

9.14.3 When a Marketing Score Crosses a Regulatory Line

The third vignette is the regulatory boundary, because propensity machinery does not stay in marketing. The same scored file that ranks customers for retention vouchers can rank them for credit-like treatments — preapproved financing offers, deposit requirements, payment-plan eligibility, service-tier routing — and the moment it does, it leaves the domain of marketing judgment and enters one governed by statute, supervision, and litigation.

The practice lesson here is a boundary discipline rather than a legal conclusion, and the reason is that the legal ground genuinely moves. In the United States, the federal treatment of disparate impact in credit was substantially revised in 2026: the Consumer Financial Protection Bureau published a final rule amending Regulation B, effective July 21, 2026, stating that the Equal Credit Opportunity Act does not authorize disparate-impact liability and removing the effects test from the regulation (Consumer Financial Protection Bureau, 2026). Commentators note that disparate-impact theories may still operate under other federal statutes and under state fair-lending law, and litigation over the rule's scope was active at the time of writing. This chapter therefore declines to state a general rule and states an obligation instead: credit-adjacent uses of customer scores can trigger federal and state fair-lending, discrimination, adverse-action, explanation, and consumer-protection requirements; the governing rules are jurisdiction- and use-specific and change; and they should be reviewed with legal and compliance specialists before deployment rather than reasoned out from a textbook paragraph.

What does not change is the mechanism the analyst is uniquely positioned to see, and it is the one Barocas and Selbst (2016) mapped and Section 6.16 previewed for segmentation: neutral-looking features such as geography, tenure, or shopping channel can carry protected characteristics by correlation, so a model that never sees a protected attribute can still distribute its errors unequally across groups defined by one. Mature organizations institutionalize the boundary exactly the way this guide's permitted-use note does: the score's approved uses are written down, credit-adjacent repurposing triggers legal review and formal fairness testing across groups, and the analyst of record is the person expected to notice when a marketing score is drifting toward a regulated decision. The lesson: the fairness screen of Lab 9.2 Part G is not an academic garnish — it is the junior version of an audit that, in adjacent industries, is performed under subpoena.

9.14.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into one expectation, and it is this chapter's closing thread: threshold-setting is where the analyst meets the CFO. The model will be a commodity — every platform ships one, every assistant drafts one — and the meetings that matter will be about the numbers around it: what the label means, what the two errors cost, whether the probabilities mean what they say, where the curve stops paying, what the program may and may not be used for, and whether anyone built the control group that can prove the treatment works. Every one of those is a conversation this chapter equipped, and none of them is a modeling conversation. The analysts who advance are the ones who can hold the room through the cost arithmetic — because the threshold is the one number in the pipeline that the CFO, once shown, will insist on owning, and the analyst who taught the CFO to own it becomes the person the CFO calls before every scored program thereafter.

9.15 Ethics, Fairness Across Segments and the Loops Models Close

The Business Analytics in Practice section ended at a legal boundary; this section examines the ethical ground on both sides of it, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), causal language (Section 7.15), and acting on predictions about people (Section 8.15) — to what classification adds: verdicts about individuals, sorted into groups, at scale, by machinery whose errors are invisible one customer at a time. Two obligations are new here, and the dataset was designed to make the first one unmissable.

The first obligation is fairness across segments, and its instrument is one this chapter already built: the confusion matrix, computed *by group*. A model's aggregate metrics — the leaderboard AUC, the overall precision and recall — are averages across the file, and averages hide people; that was Chapter 5's ethics, and it returns here with teeth, because a classifier can post excellent aggregate numbers while distributing its errors unequally across identifiable groups: flagging one group's loyal customers as churners at twice the rate it does another's, or missing one group's departing customers systematically. The new-store exhibit of Lab 9.2 Part G is the designed demonstration — the second strike of the dataset's ethics trap, and the sharper one. The first strike, in Chapter 6, was about deliberate differential treatment of known segments; this strike is about *accidental* differential judgment of a group nobody defined: customers of the newest stores, whose short tenure and thin history are artifacts of their stores' opening dates and StyleCraft's own under-marketing during ramp-up, and whom a naive churn model therefore scores as high-risk for reasons that describe the company's history, not the customers' intentions. A retention program acting on the raw scores would flood the newest markets with discount vouchers — training precisely the wrong price expectations in the expansion's youngest relationships — and, worse, the same scores repurposed upstream ("the new markets show terrible predicted retention") would argue for abandoning stores whose only sin is being new. The dataset's designers built the trap so that the correction is achievable with this guide's own tools: the tenure-restricted comparison of Code 9.18, the store-cohort controls the exercises rehearse, and the standing rule that the deliverable's fairness appendix reports error rates by segment, by metro, and by store cohort before any scored program launches. The disparate-impact frame of Barocas and Selbst (2016), cited since Section 1.13, names the general mechanism: models trained on data shaped by past decisions re-encode those decisions as predictions, and the analyst who does not look for the re-encoding has, by default, endorsed it.

The second obligation has no descriptive-analytics ancestor, because it belongs to models that act: the feedback loop, in which the model helps cause the outcome it predicts, and its own future training data records the episode as vindication. The mechanisms are concrete and none requires malice. A customer flagged as likely to churn is routed to reduced marketing investment — why spend on the leaving? — and, starved of contact, drifts away: the prediction manufactured its own confirmation, and next year's training data files her under "correctly predicted churner," making next year's model more confident in the judgment that caused the loss (O'Neil, 2016). The retention offer itself loops in the other direction: treated customers who stay are recorded as stayers, so the treatment contaminates the label, and a model retrained naively on post-program data learns that high-risk profiles stay — because the program kept saving them — degrading exactly where the program worked; the repair, at concept level, is to train on untreated customers or on the control group the Section 9.13 deliverable insisted on, and the full machinery is one more argument for randomized holdouts as standing infrastructure rather than occasional experiments. Even the churn-increasing intervention of Section 9.14 is a feedback loop in miniature: the model's list determined who received the contact that changed the outcome the model was predicting. The general statement deserves the chapter's last box, because it is the deepest way this chapter's subject differs from every one before it: a descriptive summary leaves the world as it found it; a deployed classifier is an *intervention wearing an observation's clothes*, and the analyst who deploys one inherits a monitoring obligation — not just "is the model still accurate" (Section 8.10's decay) but "is the model changing the population it scores," which no accuracy metric detects.

CONCEPT

The Second Strike, and the Loop

The guide's running ethics thread has moved from data (what may be collected), through framing and measurement (what gets asked and counted), to summaries and segments (who gets averaged and who gets grouped), to predictions (who gets acted on). Classification completes the arc: verdicts, at scale, that can differ unfairly across groups and can cause what they claim to foresee.

> The two standing instruments this chapter adds to the deliverable — the fairness screen (error rates by group, with artifact checks like the tenure-restricted new-store comparison) and the control group (the only reliable detector of a model's own causal footprint) — cost little and are, in this guide's judgment, non-negotiable for any scored program that touches customers. Neither is a certification: passing an error-parity screen does not establish that a program is fair, and a control group measures the treatment's effect rather than its ethics. What they establish is that someone looked, on the record, before launch. The analyst of record signs the scores; from this chapter forward, the signature covers not only what the model predicts but what the program does to the people predicted about.
>
> Source: Course concept developed for this guide, informed by Barocas and Selbst (2016) and O'Neil (2016).

9.16 Chapter Summary

This chapter completed the predictive machinery Part II has been assembling, by carrying it across the line from how much to whether. The frame, the label wall, the leakage audit, the training-only selection protocol, the sealed test set, and the fixed leaderboard all transferred from Chapter 8 intact; what changed was the geometry of being wrong. A binary decision fails in two directions with two prices, and the chapter's whole architecture followed from refusing to average those prices: the confusion matrix kept the four outcomes separate and named them in program vocabulary, with its cost column priced against the program's own ledger rather than against the full margin of a lost customer; precision and recall gave the two stakeholders their two questions; accuracy was demoted to a number reportable only beside its base rate and the do-nothing floor, with the margin between them computed rather than implied; AUC graded the ordering the scorers produce, on its full zero-to-one range, with ties counted as half and with its silences stated — and was assigned the job it can do, choosing a model family, rather than the job it cannot, pricing a program; and calibration was promoted from a footnote to a gate, because the chapter's destination multiplies probabilities by dollars and a well-ordered score can still be the wrong ruler.

The chapter's destination redeemed a seven-chapter-old promise: the asymmetric cost of being wrong, introduced in Chapter 2 as interpretive discipline and previewed in Chapter 8 as expected-value framing, became the decision threshold — twelve dollars against forty, a cut at 0.30 instead of the default's 0.5, derived as arithmetic a CFO can re-run, shown with the sensitivity that reveals whether the model or the economics is the binding uncertainty, and applied only after a reliability exhibit established that the probabilities could bear it. Around that center, the chapter built the working instruments of scored marketing: the churn label as a measurement decision with a policy inside it, defined from purchase activity rather than from net revenue and carrying both of Chapter 8's eligibility conditions; logistic regression reading risk in odds, with trees and neighbors as structurally different second opinions whose flexibility was chosen inside the training folds rather than declared; and the behavioral propensity exhibits — deciles, lift, gains — that turn a scored file into a budget argument with a stopping rule, carefully distinguished from the treatment-assignment propensity score Chapter 11 will need.

The AI section extended the five-point audit with the classification supplement — base rate and floor, matrix arithmetic by hand, threshold provenance, calibration and selection hygiene — and the labs ran the whole discipline twice: once at hand scale on ten customers where three policies tied at exactly 0.70 accuracy while netting nothing, thirty-two dollars, and forty-eight dollars — and where the incumbent rule, tied with none of them, lost eight — and once at full scale across eighteen numbered cells, where flexibility was chosen by a one-standard-error rule, the family was chosen on ranking evidence alone, the selected model was calibrated inside the training data and made to clear an economic gate before any money was discussed, and the deliberate failures were manufactured and caught without ever spending the test set on a diagnostic. The interpretation, practice, and ethics sections carried the scores into the rooms that matter: the platform flag met the supplement's questions, the probability was defended from becoming a verdict, the threshold was defended from being applied to unchecked scores, industry's own scar tissue was read at market prices — with the churn tournament's actual finding restored, the two Ascarza results separated, and the 2026 fair-lending change treated as a moving legal boundary rather than a settled rule — and the deliverable

acquired its two standing appendices: the fairness screen that catches a model judging customers for their stores' age, and the control group that catches a model causing what it predicts.

Looking ahead, the machinery now predicts individuals — their spend, their departure — from a snapshot of their history. The next family of marketing questions abandons the individual for the aggregate and the snapshot for the flow of time itself: what will revenue be in December, how will the new stores' ramp reshape the season, what should the buy plan assume about demand that has not happened yet? Time changes the rules more deeply than the target type did. Observations are no longer exchangeable customers but ordered days, the random splits this Part has relied on would let the future leak into the past wholesale, and the baselines with teeth stop being spreadsheets and become the calendar's own rhymes — tomorrow resembles today, December resembles last December. The next chapter takes up forecasting demand, sales, and campaign performance: trend, seasonality, and the discipline of predicting a future the model must be graded against honestly — including the one event StyleCraft's own history planted in the series, the expansion's inflection, which any forecast worth funding has to see coming.

9.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: define the label in writing before touching data, carry the base rate into every exhibit, verify confusion-matrix arithmetic by hand, choose model settings inside the training folds, check calibration before computing dollars, derive thresholds from stated costs rather than inheriting them, grade rankings with lift and gains against real incumbents, audit AI-built classifiers with the supplement, and check every scored program's behavior across customer groups. They are organized into three groups. Core chapter practice is the required path and should be completed by every student; it holds the two homework submissions from which your instructor will assign a subset, and both serve as direct preparation for Project #1, whose brief is released with this session and whose rubric appears in Appendix F. In-class activities are prepared for discussion rather than submitted. Extensions are optional. Each exercise also carries its assignment label so that instructors can assign selectively.

9.17.1 Core Chapter Practice

Exercise 9.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- Distinguish classification from class-probability estimation, and explain why this guide insists the threshold that converts one into the other is not a model property.
- Explain why churn must be *defined* rather than observed in a non-contractual business, and name the three definitional levers of Table 9.1 with one consequence each.
- The chapter builds the churn label from counted purchase orders rather than from zero net revenue. Give two concrete ways the two definitions can disagree, and state which one the deliverable should report.
- State both conditions of this chapter's eligibility rule, and explain what breaks in the feature table if the second condition is dropped.
- A logistic coefficient on discount_share is 1.1, exponentiating to about 3.0. Write the honest plain-language sentence, including the ceteris paribus clause and a verb that survives Section 7.2.
- Why does k-NN require the scaling discipline of Section 6.7 when logistic regression does not require it for validity?
- A churn model posts 88 percent accuracy on a file whose base rate is 9 percent. Compute the majority-class baseline, state the margin, and state what the comparison does and does not establish.
- State precision's question and recall's question in Comeback Edit vocabulary, name the stakeholder who owns each, and explain why the threshold moves them in opposite directions.
- AUC is 0.77. Write the one-sentence probabilistic interpretation including the treatment of ties, state the full range AUC can occupy and what a value below 0.5 would mean, and name two things this number does not certify.

Define calibration, explain how it differs from discrimination, and state precisely which step of the Section 9.9 derivation fails when scores are well ordered but systematically too high.

An analyst inherits a confusion matrix cut at 0.5 and is told "that's just the standard." State the cost assumption the default embodies, name the two further conditions that ride inside the equal-cost reading, and show with this chapter's ledger what threshold the program's own economics imply.

Distinguish the behavioral propensity score of this chapter from the treatment-assignment propensity score of causal inference, and give one sentence on why confusing them would matter in Chapter 11.

Exercise 9.2 Confusion-Matrix and Policy Arithmetic (Required Practice)

Using only the ten-customer miniature of Lab 9.1 Part A (no code): rebuild both confusion matrices — thresholds 0.50 and 0.30 — customer by customer, and verify every figure in the Verification Check: the cell counts, both accuracies, both precisions, both recalls, and the F1 at each threshold (approximately 0.57 and 0.67; show the harmonic-mean arithmetic). Then rebuild the five-policy economics table — treat nobody, the 0.50 cut, the 0.30 cut, the ninety-day rule, and the blanket — and verify the nets of $0, +$32, +$48, −$8, and $0. State explicitly what your table does with precision in the do-nothing row and why that is not zero. Identify the single customer whose reclassification between the two thresholds contributes most to the net improvement, and write one sentence on what her profile says about which customers live near a threshold.

Exercise 9.3 Derive the Threshold (Required Practice)

For each program below, derive the economic threshold from the stated costs, state the treatment rule in one sentence, and note anything the arithmetic reveals about the program's design. Close each with one sentence on what calibration evidence you would require before applying your threshold to a model's raw scores.

A win-back email costs $3 per treated customer once redemptions and margin give-up are blended, against a $30 expected benefit per treated true churner. Derive the threshold and explain what a very low threshold implies about who should receive the email — and what Section 9.14's first vignette warns even "nearly free" treatments can cost.

A high-touch save program — a personal stylist call plus a $40 credit — costs $55 per treated customer, against an expected benefit of $110 per treated true churner. Derive the threshold and state, using the gains-chart logic of Section 9.10, why this program and the email program should not share a treated list.

A program's benefit estimate is uncertain: finance brackets it between $24 and $72, against a $12 treatment cost. Derive the threshold at both ends, and write two sentences on what the range means for the deliverable, per Section 9.9's sensitivity discipline.

Exercise 9.4 Spot the Failure (Required Practice)

Each scenario below contains at least one failure from this chapter (or none). Name it, cite the section or table row, and state the repair.

A churn model's features include days_since_last_purchase computed at the June 30, 2026 analysis date, for a label whose outcome window began January 1, 2026. Its test AUC is 0.96.

A team fits logistic regression, a tree, and a k-NN, compares all three on test AUC, then tries three tree depths and keeps the one with the best test AUC. Everything else in the pipeline is clean.

An assistant, told the classes are imbalanced, oversamples churners until the training data is 50/50, then computes an expected-value table at the 0.30 threshold from the resulting scores.

A deck reports precision of 0.81 and recall of 0.74 for the same model. A footnote reveals the two figures were achieved at different thresholds.

A team evaluates its churn model with a stratified 80/20 split, selects model family and settings by stratified cross-validation inside the training customers, reports test AUC beside the majority-class baseline and the incumbent rule, and prints an out-of-fold reliability table beside its threshold derivation.

A churn model's AUC is 0.79 and its scores near the economic threshold run about fifteen percentage points above the observed churn rate in those bins. The team applies the 0.30 cut anyway, on the grounds that AUC is strong.

A retention program treats the top decile, and the following year's retrained model shows the old top decile's profiles now churn at nearly the base rate. The team concludes the original model was wrong.

A model's aggregate recall is 0.72, but recall computed within the resort-metro cohort is 0.31. The deck reports only the aggregate.

Exercise 9.5 The Full Churn Deliverable (Homework Submission)

Complete Labs 9.1 and 9.2 on the certified files and assemble the targeting deliverable per Section 9.13: the label definition page with its activity basis, both eligibility conditions, and the rejected alternatives; the training-only leaderboard with three models, the constant-score floor, and the continuous-recency baseline on cross-validated AUC and expected net value, with the base rate stated and the paired fold margins shown; the declared selection rule and the single sealed-test grade; the threshold derivation with its sensitivity table and its out-of-fold reliability exhibit; the hand-reconciled five-policy table at the derived threshold, including the ninety-day rule and the blanket, with the precision-expectation paragraph for the executive reader; the ROC, lift, and gains figures with the incumbent marked as an operating point; the budget sentence the stopping rule produces; the incremental caveat with the control-group recommendation; the monitoring plan including base-rate and calibration monitoring; the permitted-use note; and the fairness appendix with per-group error rates, the new-store exhibit, its tenure-restricted comparison with the residual tenure imbalance reported, and the sentence framing it as a minimum diagnostic screen rather than a certification. Submissions are graded on the threshold derivation, the calibration evidence, and the fairness appendix as heavily as on the models — and this deliverable's structure is the template for Project #1's predictive track (Appendix F).

Exercise 9.6 AI Classification Audit (Homework Submission)

Give an AI assistant the raw StyleCraft tables and this deliberately loose prompt: "Build me the best churn model you can — maximize accuracy." Audit the response in writing with Table 8.5's five points followed by Table 9.5's four: reconstruct the churn definition the assistant invented and compare it against Section 9.3's, naming which levers of Table 9.1 it moved and whether it built the label from activity or from revenue; trace every feature across the snapshot, running the recency range check; identify the threshold it applied and whether it disclosed applying one; compute the base rate and majority-class baseline it omitted and state the margin; determine whether any model family or setting was chosen by consulting a test split; note whether any calibration evidence accompanies any dollar figure; and re-derive its headline metrics from its own confusion matrix, reconciling the cell counts by hand. Then re-prompt with the full frame per the AI in Practice box and compare the two responses' leaderboards. Document both exchanges per the AI-use documentation template in Appendix D, and conclude with two sentences on which supplement point caught the most serious problem.

9.17.2 In-Class Activities

Exercise 9.7 The Fairness Slice (In-Class Discussion)

Your team will prepare the fairness appendix conversation for the Comeback Edit launch meeting. Prepare: the new-store exhibit from Code 9.18, with the tenure-restricted comparison and a one-paragraph explanation, in Section 7.2's verbs, of why the raw score gap is largely an artifact; per-group error rates (by metro and by lifecycle group) at the 0.30 threshold, identifying any group whose false-positive or false-negative rate departs materially from the aggregate and reporting every group's size beside its rates; a recommendation on what the program should do differently, if anything, for new-store customers — with the trade-offs stated, since both over-treating and excluding them have costs; and a position on the question the meeting will actually ask: "the model is accurate overall — why are we complicating the launch?" Half the class prepares that skeptical position seriously; the strongest version of the skeptic's case is what the fairness appendix must be built to survive. Close by stating, in one sentence, what your screen does *not* establish.

Exercise 9.8 Find the Flaw in the Scored Program (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

A retention program celebrates its first quarter: 60 percent of treated customers stayed. No control group was held out, and the team cannot say what fraction would have stayed untreated.

A vendor's churn module is adopted because its 0.94 AUC beat the in-house model's 0.77. The vendor's features are proprietary, and its evaluation data cannot be inspected.

The Comeback Edit list is cut at the budget's 1,600-customer capacity. Nobody has computed where the economic threshold falls, so nobody knows whether the budget is too small, too large, or right.

A team tunes its classifier to maximize F1, presents the result as "optimal," and cannot say what the program's false-positive and false-negative costs are.

A deck reports that the model's top decile has a lift of 1.05 and recommends funding the top two deciles because "the model beats random."

After a successful year, the churn scores are adopted by the finance team to set deposit requirements for a new payment-plan product. No new review occurs, because "the model is already validated."

A model predicts churn for the newest cohort at twice the file's rate. The expansion review deck cites this as evidence the new markets are failing. State the two distinct analyses from this chapter that the deck skipped.

9.17.3 Extensions

Exercise 9.9 Customer-Specific Thresholds (Optional)

Section 9.9 assumed one benefit figure for every customer, and Chapter 5's concentration findings guarantee that assumption is false. Using the scored test file, replace the single $160 retained-margin figure with a customer-specific estimate — the customer's own feature-window revenue_per_month projected across the planning horizon is a defensible starting point, and you should defend whatever you choose — then derive a customer-specific threshold for each customer and recompute the five-policy table with the personalized rule added as a sixth row. Report three things: how the treated-list size changes, how the expected net changes, and which customers enter or leave the list. Then write two paragraphs on what this refinement costs. One should address measurement (the personalized benefit is itself an estimate with its own error, and the threshold now inherits two uncertainties instead of one); the other should address governance (a program in which different customers face different treatment thresholds is harder to explain, harder to audit for the group effects of Section 9.15, and harder to defend if the score is ever repurposed).

9.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Accuracy. The share of evaluated cases whose predicted class matches the actual class; under class imbalance, dominated by the majority class and reportable only beside the base rate and the majority-class baseline, with the margin between them computed (adapted from Provost & Fawcett, 2013).

AUC (area under the ROC curve). A threshold-free summary of ranking skill ranging from 0 to 1, equal to the probability that a randomly chosen positive case receives a higher score than a randomly chosen negative one, with ties counted as half. A value of 0.5 indicates chance-level ordering and values below 0.5 indicate a systematically reversed or worse-than-chance ordering; the measure is silent about calibration and costs (adapted from Fawcett, 2006; Google for Developers, 2026).

Base rate. The share of the positive class in the evaluated population; the context number without which no classification metric is legible, and itself a constructed quantity under a constructed label (adapted from Provost & Fawcett, 2013).

Behavioral propensity score. A model-estimated probability of a defined customer behavior, used to rank customers for differential treatment; the packaged form in which this chapter's machinery appears throughout the marketing stack, and distinct from the treatment-assignment propensity score of causal inference (adapted from Provost & Fawcett, 2013).

Calibration. The property of a probability scorer whose predicted probabilities match observed event frequencies, so that among cases scored near p about a share p are positive; distinct from discrimination, assessed with a reliability exhibit of mean predicted probability against observed rate by score bin, and required before an economically derived threshold may be applied to raw scores (adapted from scikit-learn developers, 2026a).

Churn label. The operationalized record of churn for a historical case: a defined activity — here, at least one completed purchase order — failing to occur within a defined outcome window after the snapshot, under stated eligibility rules; a measurement decision whose definitional levers change the model and the treated population (adapted from Neslin et al., 2006; Provost & Fawcett, 2013).

Class imbalance. The condition in which one class heavily outnumbers the other, under which accuracy is dominated by the majority class and rank-based or cost-based evaluation is required (adapted from He & Garcia, 2009).

Class-probability estimation. The prediction, for each case, of the probability of the positive class — a score that ranks cases and defers verdicts to a separately chosen threshold (adapted from Provost & Fawcett, 2013).

Classification. The prediction of a categorical target by assigning each case to a class; in this guide, produced by thresholding estimated probabilities rather than emitted directly by the model (adapted from Provost & Fawcett, 2013).

Classification tree. A model that predicts by routing cases through a learned sequence of yes/no feature questions to leaves, where predicted probabilities are the leaves' training-class shares; grown by recursive splitting and governed by depth or leaf-size limits chosen inside the training data (adapted from Breiman et al., 1984).

Confusion matrix. The cross-tabulation of actual against predicted classes at a stated threshold, partitioning cases into true positives, false positives, false negatives, and true negatives; the arithmetic source of every threshold-dependent metric (adapted from Provost & Fawcett, 2013; Fawcett, 2006).

Decision threshold. The probability above which a scored case is treated as positive; a policy choice allocating errors between false positives and false negatives, derived from the two errors' relative costs rather than from the software default of 0.5, and valid only where the scores are sufficiently calibrated (adapted from Provost & Fawcett, 2013; scikit-learn developers, 2026b).

F1 score. The harmonic mean of precision and recall, dragged toward the smaller of the two; a single-number summary that weights the two errors equally and is therefore provisional wherever costs are asymmetric (adapted from Provost & Fawcett, 2013).

False negative. A positive case predicted negative — here, a departing customer no one tried to save; priced as the missed expected incremental save rather than as the customer's full future margin (adapted from Provost & Fawcett, 2013).

False positive. A negative case predicted positive — here, a retention offer spent on a customer who was staying; priced as the blended treatment cost, and bought deliberately by low thresholds (adapted from Provost & Fawcett, 2013).

Gains chart. The cumulative share of all positive cases captured as treatment extends down the score-ranked file, plotted against the share of the file treated; read against the random-targeting diagonal and used with treatment economics as a budget stopping rule (adapted from Provost & Fawcett, 2013; Neslin et al., 2006).

k-nearest neighbors (k-NN). A model that predicts a case's class probability as the positive share among its k closest training cases in scaled feature space; coefficient-free, with flexibility governed by k and k chosen by cross-validation inside the training data (adapted from Cover & Hart, 1967).

Lift. The ratio of a targeted group's positive rate to the base rate — how many times richer in the target class a model-selected slice is than a random one; the ranked-targeting metric with its baseline built in (adapted from Provost & Fawcett, 2013).

Log-odds. The natural logarithm of the odds; the unbounded, symmetric rescaling of probability on which logistic regression's linear equation operates (adapted from Hosmer et al., 2013).

Logistic function. The S-shaped curve that converts any log-odds value into a probability between 0 and 1 — flat near the extremes, steepest at even odds (adapted from Hosmer et al., 2013).

Logistic regression. The model that expresses the log-odds of a binary target as a linear function of features and recovers probabilities through the logistic function; its exponentiated coefficients are read as multiplicative effects on odds, ceteris paribus (adapted from Hosmer et al., 2013; James et al., 2021).

Odds. The ratio of an event's probability to its complement's — $p \div (1 - p)$; the currency in which logistic coefficients speak after exponentiation (adapted from Hosmer et al., 2013).

Precision. The share of predicted positives that are actually positive — the purity of the treated list, and finance's question; undefined rather than zero when nothing is treated (adapted from Provost & Fawcett, 2013).

Recall (true positive rate). The share of actual positives predicted positive — the coverage of the at-risk class, and the churn-number owner's question (adapted from Provost & Fawcett, 2013; Fawcett, 2006).

ROC curve. The plot of true positive rate against false positive rate across all thresholds, tracing a scorer's ranking trade-off independent of any single cut (adapted from Fawcett, 2006).

True negative. A negative case predicted negative — the quiet majority correctly left untreated, and the cell that inflates accuracy under imbalance (adapted from Provost & Fawcett, 2013).

True positive. A positive case predicted positive — here, a save opportunity reached (adapted from Provost & Fawcett, 2013).

9.19 Further Readings

Students who want additional background may begin with the following readings. The business-first treatments are listed first, methods second, software and consequences last.

Provost and Fawcett (2013), especially the chapters on model evaluation, expected value, and ranking (lift and profit curves), for the business-first development of nearly everything this chapter built — the closest published relative of its approach, as it was for Chapter 8.

Neslin et al. (2006) for the churn-modeling tournament that graded forty-five submissions on common validation data and found that modeling approaches differ materially in predictive and economic performance — the empirical case for this chapter's leaderboard discipline, and a useful corrective to the folk claim that method choice does not matter.

Fawcett (2006) for the standard accessible introduction to ROC analysis — the extended companion to Section 9.8, with the practitioner's cautions included.

Ascarza (2018) and Ascarza et al. (2016), read as a pair: the first demonstrates that targeting by churn risk can misallocate retention because risk is not responsiveness, and the second demonstrates, in a field experiment, that a proactive retention campaign can raise churn outright. Together they are the cleanest bridge from this chapter to Chapter 11.

James et al. (2021), Chapter 4, for the standard development of logistic regression, k-NN, and classification evaluation, one level of formality above this chapter — and, for the implementation of exactly these ideas, the three pages of official scikit-learn documentation these labs are built on (scikit-learn developers, 2026a, 2026b, 2026c): probability calibration, post-hoc threshold tuning, and the cross-validation guidance that keeps development inside the training folds.

O'Neil (2016) for the general-audience treatment of feedback loops and scaled algorithmic harm — the accessible companion to Section 9.15's second obligation, read alongside Barocas and Selbst (2016) for the scholarly treatment of the disparate-impact mechanism itself.

9.20 References

Ascarza, E. (2018). Retention futility: Targeting high-risk customers might be ineffective. *Journal of Marketing Research, 55*(1), 80–98. https://doi.org/10.1509/jmr.16.0163

Ascarza, E., Iyengar, R., & Schleicher, M. (2016). The perils of proactive churn prevention using plan recommendations: Evidence from a field experiment. *Journal of Marketing Research, 53*(1), 46–60. https://doi.org/10.1509/jmr.13.0483

Barocas, S., & Selbst, A. D. (2016). Big data's disparate impact. *California Law Review, 104*(3), 671–732. https://doi.org/10.15779/Z38BG31

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Wadsworth.

Consumer Financial Protection Bureau. (2026). *Equal Credit Opportunity Act (Regulation B)*, 12 C.F.R. pt. 1002. https://www.consumerfinance.gov/rules-policy/regulations/1002/

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*(8), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Google for Developers. (2026). *Classification: ROC and AUC* [Machine Learning Crash Course]. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering, 21*(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). Wiley. https://doi.org/10.1002/9781118548387

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Neslin, S. A., Gupta, S., Kamakura, W., Lu, J., & Mason, C. H. (2006). Defection detection: Measuring and understanding the predictive accuracy of customer churn models. *Journal of Marketing Research, 43*(2), 204–211. https://doi.org/10.1509/jmkr.43.2.204

O'Neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

scikit-learn developers. (2026a). *Probability calibration* [Software documentation]. https://scikit-learn.org/stable/modules/calibration.html

scikit-learn developers. (2026b). *Post-hoc tuning the cut-off point of decision function* [Software documentation]. https://scikit-learn.org/stable/auto_examples/model_selection/plot_tuned_decision_threshold.html

scikit-learn developers. (2026c). *Cross-validation: Evaluating estimator performance* [Software documentation]. https://scikit-learn.org/stable/modules/cross_validation.html

Chapter 10

Forecasting Demand, Sales, and Campaign Performance

From Customers to Calendars

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter develops forecasting as the time-indexed branch of the predictive discipline built in Chapters 8 and 9, with StyleCraft's holiday open-to-buy plan as its running decision. It establishes the time series as a data structure whose order carries information, uses decomposition to read trend, seasonality, cycle, and noise, and separates recurring calendar effects from structural breaks. It builds the forecasting ladder from the naive and seasonal-naive opponents through the exponential smoothing family, then rebuilds honest evaluation for ordered data: rolling-origin selection, a sealed holdout that resembles the decision, MAPE with its denominator failure, and accuracy reported by horizon. The forecast widens into a distribution through simulated paths and scenarios, and the human hand is disciplined by the written override. The chapter closes with managerial interpretation, industry practice, the ethics of the laundered guess, and exercises.

KEYWORDS

forecasting; time series; seasonality; structural break; seasonal-naive; exponential smoothing; MAPE; prediction interval; forecast horizon; judgmental adjustment

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Forecasting demand, sales, and campaign performance. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 10, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, NumPy, Matplotlib, and scikit-learn are sponsored or affiliated projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. statsmodels is a community-developed project distributed under the modified BSD license. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: Applied Business Analytics companion repository on GitHub

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and

recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

22. Explain what makes a time series a distinct data structure — ordered observations whose sequence carries information — and construct a daily revenue series from transaction-grain data by a deliberate grain change, per Section 3.3 and the groupby machinery of Chapter 4.
23. Name the four components of a marketing time series — trend, seasonality, cycle, and noise — read each from a decomposition plot at concept level, and state which component each forecasting method in this chapter is built to track.
24. Distinguish recurring calendar effects from structural breaks, explain why history from before a break misleads a model after it, and identify StyleCraft's store-opening inflection as a designed break announced in the company's own data.
25. State the boundary between the weekly seasonality a period-7 method carries and the annual, moving, and promotional calendar effects it does not, and name the three instruments that carry the second group.
26. Construct naive and seasonal-naive forecasts as the time extension of the baselines of Section 8.7, and defend the seasonal-naive rule as the opponent every forecasting model must beat before it earns a planning decision.
27. Describe the exponential smoothing family at working level — simple smoothing, Holt's trend method, damped trend, and Holt-Winters seasonal smoothing — and state what each added layer tracks, what it costs, and how each can overfit.
28. Explain why random train/test splits fail for ordered data, and build the two-layer evaluation this chapter requires: rolling-origin comparison inside the training data for method selection, and one sealed final holdout, chosen to resemble the decision, for the grade.
29. Compute and interpret forecast accuracy with MAPE alongside the MAE and RMSE of Section 8.6, guard MAPE's denominator in code, and report accuracy as a function of horizon estimated across several forecast origins rather than one.
30. Distinguish a point forecast from the distribution it summarizes, build daily, weekly, and season-total prediction intervals by aggregating simulated paths before taking quantiles, and state what a simulation band does and does not price.
31. State when a judgmental adjustment to a statistical forecast is legitimate, size an override as a ramp from controlled evidence with a stated range, and apply the six-element written-override discipline.
32. Recognize the three standing forecasting failure modes — extrapolating through structural breaks, overfitting seasonality, and ignoring promotions — and diagnose each from a forecast exhibit.
33. Verify an AI-drafted forecasting pipeline with the five-point audit of Section 8.11 plus the forecasting supplement, predicting the holdout error before running, and documenting the audit per Appendix D.
34. Assemble a forecast deliverable a planning meeting can hold accountable: method and opponent stated, sealed-holdout grade in MAPE and dollars, intervals communicated in words, overrides documented, and the forecast translated into a declared planning unit whose limits are stated.

Chapter 9 ended with a scored file and a signature. The Comeback Edit's targeting rule was locked — the churn label defined in writing, the threshold re-derived from the program's own twelve-dollar economics, the fairness appendix catching the model before it condemned the newest stores for being new — and the analyst of record signed a deliverable that made predictions about individual people and owned what the program would do to them. Every prediction in that deliverable shared a quiet structural assumption: the customers were exchangeable. The split shuffled them freely, the model scored them in any order, and nothing about customer C004471's row depended on

which row came before it. This chapter forfeits the assumption, because the next family of marketing questions abandons the individual for the aggregate and the snapshot for the flow of time itself. What will revenue be in December? How will the new stores' ramp reshape the season? What should the buy plan assume about demand that has not happened yet? The observations are no longer eight thousand interchangeable customers but roughly seven hundred and thirty ordered days, each one glued to its neighbors by trend, rhythm, and event — and the ordering is not an inconvenience to be shuffled away but the single most informative thing about the data. The machinery of Part II rides along where it can: the frame is still declared before fitting, the baseline is still named before the model exists, the test data is still the future the model has not seen. But time rewrites the mechanics of every one of those disciplines, adds two instruments the customer-grain chapters never needed — the decomposition that separates a series into its stories, and the prediction interval that keeps a forecast honest — and installs this chapter's hardest lesson at the exact place StyleCraft's own history planted it: a model trained on the past can only extend the past, and when the business has deliberately broken with its past — an expansion, an opening, a new market — the break is something the analyst must bring to the model, because no amount of fitting will make the model see it on its own.

CONCEPT

What This Chapter Is Really About

Chapters 8 and 9 graded models against a future made of held-out customers — a future the analyst could manufacture at will, because customers are exchangeable and any random twenty percent of them stands in for the rest. Time offers no such courtesy. The future of a time series is not a random sample of its past; it is the next segment of a single unrepeatable path, connected to everything before it and different from all of it. That one fact rearranges the chapter. Evaluation must move the wall of Section 8.3 from between customers to between yesterday and tomorrow, because a shuffled split lets the model interpolate — predict a missing Tuesday from the Tuesdays on either side of it — which is a party trick, not a forecast. The baseline must learn the calendar, because "tomorrow resembles today" is free, obvious, and brutally hard to beat. And the model must be caught at the boundary of its competence: every method in this chapter works by extending patterns the history contains, which means every method is structurally blind to the two things StyleCraft's planning team most needs it to see — the deliberate break with history that the expansion is, and the annual holiday shape that two Decembers cannot teach it.

The chapter's real subject is the resulting division of labor. The model owns the historical rhymes: level, trend, and the weekly rhythm. The analyst owns the future calendar the model cannot read: the announced opening, the moving holiday, the planned promotion, the future the company intends. And the forecast the meeting can trust is the one where both contributions are visible, graded, and signed — the model's on a holdout chosen to resemble the decision, the analyst's in a written override.

Source: Course concept developed for this guide, informed by Hyndman and Athanasopoulos (2021) and Fildes et al. (2009).

10.1 Marketing Decision Context: The Buy Plan That Assumes a Future

Every deliverable in Part II so far has predicted people. The Backstage list ranked customers by expected spend; the Comeback Edit ranked them by risk of leaving. The decision that opens this chapter predicts no one in particular — and commits more money than both programs combined. It is late summer at StyleCraft, and the head of merchandise planning owns the two documents that will define the company's fourth quarter. The first is the holiday open-to-buy plan: the dollar envelope of receipts the November–December assortment may commit, which must be signed roughly six weeks from now because apparel lead times are long and the factories' calendars do not negotiate.

The second is the Q4 media pacing plan: how the season's marketing budget spreads across the weeks from late October through the December peak. Both documents assume a future. An open-to-buy envelope is a bet on December demand placed in August; a pacing plan is a bet on which weeks the demand will arrive in. And the bet is bigger and stranger this year than it has ever been, because the number both plans need — expected revenue, week by week, through the holiday season — must span the thing StyleCraft has spent two years doing to itself: the expansion. Eight Wave-4 stores have opened across the trailing year, several of them recently enough that their ramp curves are still visibly climbing in the weekly numbers. Whatever last December looked like, this December will not look like it, by design.

Three forecasts are already circulating, and — the pattern is by now the guide's oldest running joke — each arrives confident and none is accountable. The first is finance's trendline: last year's holiday revenue, grown by the year-over-year rate of the spring, drawn forward as a straight line. It is hopeful in the precise sense that it assumes the future will be the past plus a constant, and it makes no statement about what would have to be true for that to hold. The second lives inside the planning module of StyleCraft's inventory software, which offers a Demand Forecast button and, when pressed, produces a number — a single number, to the dollar, with no interval, no stated method, and documentation that says only that it uses "advanced machine learning." The third belongs to the merchant who has been right before: the founder's longtime head of buying, whose position is that the suburban stores will "come alive for holiday gifting" and that the season will be up thirty percent. She may well be right. But her number cannot be interrogated, cannot be graded, and — the planning lead's actual complaint — cannot be defended to the CFO when the receipts it justified become markdowns in January. Three numbers, three provenances, and not one of them can answer the only question that matters about a forecast before you bet on it: how wrong does this method tend to be, and how would we know?

So the planning lead has borrowed the VP of Marketing's analyst — still you — and commissioned the deliverable this chapter builds. Not a number: a forecast, in this guide's full sense. A stated method, fitted to the daily revenue history that StyleCraft's own transactions generate, and chosen by comparison rather than preference. A stated opponent — the seasonal-naive rule this chapter will introduce, which any funded model must visibly beat. A grade earned the only way a seasonal forecast can be graded honestly: by holding out a stretch of the known past that resembles the decision, forecasting it as if it were still unknown, and measuring the miss — including the test the expansion makes non-negotiable, whether the method sees the inflection the store openings printed into the series, and what it takes from the analyst to make it see the next one. Prediction intervals, because the plan needs to know not just the center of the future but its plausible width, and because a point forecast delivered without one is — Section 10.15 will use the hard word — a laundered guess. And a written override log, because the analyst knows things the history cannot: the openings already announced for the fall, the promotion calendar marketing has drafted, the moving date of Thanksgiving.

One boundary belongs in the brief rather than in a footnote nine sections later, because it constrains what the whole project can claim. This chapter forecasts chain-level revenue. A revenue forecast can size an open-to-buy envelope in dollars and pace media by week. It cannot, on its own, tell a buyer how many units of which dress in which size to send to which store, because that translation needs realized price, category and size mix, margin, markdown expectation, opening inventory, and store allocation — none of which a chain-level revenue series contains. The program is therefore commissioned as a revenue-and-pacing deliverable with a declared dollar envelope, and the unit-level assortment buy is named as a separate forecast at a separate grain, with its own baselines and its own holdout. Naming the boundary in the brief is what keeps Section 10.13 from promising quantities the analysis cannot produce.

The receipts lock in six weeks. Sections 10.2 through 10.4 teach the reader to see the series. Sections 10.5 and 10.6 build the forecasting ladder from free baselines to Holt-Winters. Sections 10.7 and 10.8 rebuild honest evaluation for ordered data and widen the forecast from a number into a distribution. Section 10.9 disciplines the human hand. Section 10.10 catalogs the failures. Section 10.11 turns to the AI assistant, which will draft any of it fluently and grade none of it honestly. The labs in Section 10.12 run the whole discipline on the certified series, and Sections 10.13 through 10.15 rehearse the meeting where a forecast becomes a commitment — and someone becomes accountable for the difference.

10.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- Finance's trendline is not stupid — it is a model, with assumptions. State the assumptions in one sentence each, and name the two features of StyleCraft's recent history (this chapter will call them a component and a break) that the trendline has no way to represent.
- The planning module's Demand Forecast button produces a single number with no interval and no method. Write the three questions you would need answered before the number could enter the plan, and note which chapters of this guide trained each question.
- The merchant's gut number cannot be graded in advance — but it can be used. This chapter will argue there is exactly one legitimate way to bring human knowledge like hers into a forecast. Predict what it is, and check your prediction against Section 10.9.
- A forecast that misses high and a forecast that misses low do different damage to a buy plan. Describe the cost of each direction in inventory terms, and name the section of this guide that taught you to expect the two costs to differ — and to say so before the analysis, not after.
- The brief separates a dollar envelope from a unit buy. State the practical consequence of that boundary for what the deliverable may hand the buying team, and what it must hand them instead.

10.2 Time Series as a Data Structure

The opening case asked for a forecast of revenue by week; this section starts further back, with the shape of the data the forecast will stand on, because the shape is the chapter's first genuinely new idea. Every dataset in Part II so far has been a table of exchangeable rows. The eight thousand customers of the feature table could be shuffled without destroying anything: no customer's row depended on which row preceded it, and the train/test split of Section 8.5 exploited exactly that freedom. A time series abandons the freedom. Its observations are indexed by time — for StyleCraft, one row per calendar day — and the index is not a label but a structure: each day's revenue is stitched to its neighbors by the trend the business is riding, the weekly rhythm of retail traffic, the season the calendar imposes, and the aftermath of whatever happened yesterday. Shuffle the rows of a time series and the object is destroyed; the information was in the order.

> **DEFINITION**
>
> **Time Series**
>
> A time series is a sequence of observations of the same quantity, recorded at successive points in time — usually at a regular interval such as a day, week, or month — in which the temporal ordering is part of the data's meaning. The order must be preserved because observations may exhibit serial dependence, trend, seasonality, or structural change, and because those are the patterns a forecast extends. Not every time series is serially dependent — a series of independent draws recorded in time order is still a time series — but the business series this chapter forecasts almost always are, and any procedure that treats their observations as exchangeable discards the temporal context needed to diagnose and forecast them.
>
> Source: Adapted from Hyndman and Athanasopoulos (2021).

In other words, time order is information — and that is the sentence that separates this chapter from the two before it. In the customer-grain chapters, knowing that one row was "next to" another told you nothing; here it tells you nearly everything, because tomorrow is constrained by today in a way that one customer never constrained another. The constraint is what a forecaster harvests: the whole enterprise of this chapter is the disciplined use of a series' own past as evidence about its future, and the methods of Sections 10.5 and 10.6 differ mainly in how much of the past they consult and how quickly they forget it.

Before any of that, the series must exist, and for StyleCraft it does not exist natively — it must be manufactured by a grain change. The company's source of truth is the transactions fact table, one row per order line, per Section 3.3's grain discipline; a daily revenue series is that table aggregated to one row per day, summing line_revenue within each order date — the deliberate grain change of Section 3.3, executed with the groupby machinery Chapter 4's labs built and this guide does not re-teach. The certified dataset ships the result as marketing_daily, the daily table that joins the transaction roll-up with campaign activity, and Code 10.3 performs the reconciliation this guide's known-truth discipline requires: rebuild the daily revenue column from the fact table yourself, and confirm the shipped series matches to the cent, because a forecast inherits every defect of the aggregation underneath it.

Two properties of the manufactured series deserve a sentence each. The grain is a choice, and it is part of the frame: the open-to-buy plan thinks in weeks, the pacing plan in days, and a series can always be aggregated upward (days to weeks) but never disaggregated downward without new assumptions — so this chapter works at the daily grain and rolls up. And the series must be complete: a day with no transactions is a revenue of zero, not a missing row, for exactly the reason a customer with no outcome-window purchases was a label of zero in Section 8.3 — absence is an observation, and dropping it would teach the model a calendar with holes in it. That rule has a mechanical consequence the lab enforces rather than assumes: reindexing a series onto a complete daily calendar inserts missing days as blanks, not as zeros, and turning the blanks into zeros is a decision the analyst makes and states, not a default the software supplies.

10.3 The Components of a Marketing Time Series

Plot StyleCraft's daily revenue across its twenty-four months and the line looks, at first, like noise with a mood. The skill this section installs is seeing the line as the sum of separable stories — because every forecasting method in this chapter is, underneath its arithmetic, a claim about which of those stories will continue. The decomposition tradition gives the stories standard names, and this guide states them once, formally, in the box below, then spends the section teaching the reader to point at each one on StyleCraft's own plot.

DEFINITION

Trend, Seasonality, and Cycle

The trend of a time series is its long-run direction — the smooth underlying path the series follows once short-run fluctuation is set aside. Seasonality is a repeating pattern with a fixed and known period: the rhythm of days of the week, or of months of the year, that recurs on schedule at a constant number of observations apart. A cycle is a rise and fall that recurs without a fixed period — economic expansions, fashion waves — distinguishable from seasonality precisely because its timing cannot be read off a calendar. What remains when trend, seasonality, and cycle are accounted for is noise (the remainder or irregular component): the unexplained residual variation no pattern claims.

Source: Adapted from Hyndman and Athanasopoulos (2021).

Each component has a StyleCraft face, and the dataset's designers planted all four deliberately. The trend is the expansion story itself: the urban business's steady organic growth, plus the stair-step contributions of each opening — a trend with a complication Section 10.4 will name. The seasonality operates at two nested periods, and the difference between them organizes the rest of the chapter. The weekly rhythm is retail's heartbeat: quiet early weekdays, a build through Friday, a weekend peak — a pattern that repeats roughly one hundred and four times in the history, which is why every method in this chapter will be given a seasonal period of seven. The annual rhythm is the merchandising calendar: the back-to-school lift and the November–December holiday surge, patterns the history contains only twice — a scarcity with consequences Sections 10.4 and 10.10 will price, and the reason this chapter never asks a period-7 method to carry them. Cycles, at this guide's concept level, are the honest admission that not every wave has a schedule: a viral product moment, a macroeconomic squeeze on discretionary apparel, a competitor's collapse. The twenty-four-month window is too short to estimate cycles and this chapter does not try; the term is owned here so the reader stops mislabeling every unexplained wave "seasonality." And the noise is

everything else — weather, whim, and the irreducible randomness of a few hundred purchase decisions a day — the component no method forecasts and every honest interval respects.

The instrument that makes the components visible is decomposition, and this guide uses it at concept level: as a reading tool, not a forecasting engine. A decomposition procedure takes the observed series and splits it into estimated trend, seasonal, and remainder parts — classical procedures do this with moving averages, and the modern standard, STL, does it with iterated local smoothing (Cleveland et al., 1990) — under an assumed structure, most simply the additive one: observed value equals trend plus seasonal effect plus remainder. The mechanics stay in the software; what the analyst owns is the reading. A decomposition plot stacks four panels — observed, trend, seasonal, remainder — and Code 10.4 will put StyleCraft's own on screen with the store-opening dates drawn on it. Read the trend panel for the expansion: the climb, and the visible steps where openings land. Read the seasonal panel for the heartbeat: the seven-day sawtooth, regular as a pulse, whose peak-to-trough height in dollars is exactly what the seasonal-naive rule of Section 10.5 carries for free. And read the remainder panel the way Chapter 4 taught you to read every residual exhibit: not as garbage but as a question list, because a spike in the remainder is a day the components could not explain — a promotion the decomposition does not know about, a storm, a data defect — and Section 10.10's third failure mode begins with an analyst who never looked.

DEFINITION

Decomposition

Decomposition is the separation of an observed time series into estimated components — trend, seasonal, and remainder (with cycle folded into trend at short horizons) — under an assumed structure, most commonly additive (the components sum to the observed value). It is used in this guide as an analytic reading instrument: a way to see which stories a series contains and how large each is, before choosing a forecasting method that must track them. A decomposition is fitted at one declared period; a decomposition at period 7 estimates the weekly pattern and folds everything annual into trend and remainder.

Source: Adapted from Cleveland et al. (1990) and Hyndman and Athanasopoulos (2021).

Table 10.1 collects the components with their StyleCraft instances and — the column that makes the table load-bearing for the rest of the chapter — the instrument that will be responsible for each. The table is a map of Sections 10.5 through 10.9 before the reader gets there: the seasonal-naive baseline handles the weekly pattern by memory, Holt's method adds a tracked trend, Holt-Winters tracks both, and the three lower rows are the ones no fitted method in the toolkit handles — the annual calendar, which needs an explicit calendar model; the break, which belongs to the analyst; and the noise, which belongs to the interval.

Table 10.1

The components of StyleCraft's daily revenue series, and what carries each

Component	StyleCraft instance	Carried by
Trend	Urban organic growth plus the expansion's cumulative lift	Holt's method and Holt-Winters (Section 10.6)
Weekly seasonality	Quiet Tuesdays, weekend peaks — period 7, about 104 repetitions in the history	Seasonal-naive baseline (Section 10.5); Holt-Winters at seasonal_periods=7 (Section 10.6)
Annual and moving calendar effects	Back-to-school lift; the November–December surge; Thanksgiving and Black Friday, whose dates move	Not carried by any period-7 method. An explicit calendar model, a prior-holiday benchmark, or documented scenarios (Sections 10.4, 10.8, 10.9)

Component	StyleCraft instance	Carried by
Promotions	Chosen-date discount events in the campaigns table; future ones in a calendar the model has never seen	The analyst, via the declared calendar and documented overrides (Sections 10.4, 10.9)
Cycle	Fashion and macro waves with no fixed period	No method here; named to prevent mislabeling
Structural break	Store-opening steps and ramps; any future opening or policy change	The analyst, via declared knowledge and sized overrides (Sections 10.4, 10.9)
Noise	Irreducible day-to-day randomness	No method; priced by the prediction interval (Section 10.8)

10.4 Calendar Effects and Structural Breaks

The seasonal component of Section 10.3 deserves a sharper look before any method is built on it, because "seasonality" in marketing data is really a bundle of calendar effects — and most members of the bundle are not carried by a seasonal period at all. Calendar effects are the systematic influences the calendar exerts on a business series: the day-of-week pattern; the fixed-date holidays; the floating retail events — Thanksgiving and the Black Friday weekend it anchors — whose dates move within a window; and paydays, school calendars, and the merchandising rhythms a drops-driven brand imposes on itself. What unites them is knowability: every one of them is already on next year's calendar. What divides them is whether a fitted seasonal component can carry them, and this is the distinction the draft of this chapter blurred and the audit of it corrected.

CONCEPT

Knowable Is Not the Same as Carried

It is tempting to reason that because calendar effects are known in advance, a method with seasonal memory will carry them. Knowable dates and repeating seasonal indices are not the same thing, and the difference decides what the analyst must supply.

A day-of-week rhythm is carried automatically by a period-7 seasonal component, because it recurs at a constant number of observations apart, every time, and a two-year daily history contains about a hundred repetitions of it. Christmas is known, but the weekday it falls on changes, so its effect on a daily series lands at a different position in the weekly cycle each year. Thanksgiving and Black Friday move within November entirely, so a day-of-year index misaligns them by up to six days. A promotion is a planned event, not a season: the company chooses the date, and next year's dates live in a marketing calendar the model has never seen. A store opening is a structural change, not a seasonal effect at all.

The working rule: calendar effects are knowable in advance, but only stable fixed-period patterns are carried automatically by seasonal memory. Moving holidays, promotions, and openings must be supplied explicitly, through calendar variables, a prior-period benchmark, documented scenarios, or written overrides. Daily retail series routinely contain more than one seasonality, and moving holiday effects are handled with explicit indicators rather than a fixed seasonal lag.

Source: Course concept developed for this guide, informed by Hyndman and Athanasopoulos (2021).

The boundary that follows is stated once here and honored everywhere after, because half the chapter's evaluation design exists to enforce it. This chapter's Holt-Winters model is fitted at seasonal_periods=7. It tracks the weekly rhythm and nothing else seasonal. StyleCraft's annual holiday demand is two observations of a 365-position pattern, one of them mid-expansion; it cannot be estimated reliably from that, and this chapter does not pretend otherwise. Annual holiday structure and moving retail events are carried by three instruments instead, all of them the analyst's:

an explicit calendar layer estimated from the prior holiday and aligned on the moving date rather than the calendar date (Code 10.9); documented scenarios that vary the calendar assumption (Code 10.20); and written overrides with owners and review dates (Section 10.9). The seasonal-naive rule does not rescue this either: at a seven-day lag it repeats last week, not last December, and a daily annual seasonal-naive rule would have to reckon with 365- and 366-day calendars and with holidays that move. The chapter does not need dynamic harmonic regression to make this point. It needs to stop implying that a period-7 model carries annual holiday structure — and then to build an evaluation that proves it does not.

The structural break is the calendar effect's dangerous cousin: an event that changes the series' underlying behavior rather than decorating it. A seasonal effect visits and leaves; a break moves in. After a break, the level, the trend, or even the seasonal pattern of the series is different — permanently, or at least indefinitely — and the history from before the break describes a process that no longer exists.

DEFINITION

Structural Break

A structural break is a point in time at which the data-generating process behind a series changes — a shift in level, trend, or seasonal pattern that persists after the event rather than reverting. History from before a break is evidence about a different regime: models fitted across an unmodeled break blend the two regimes into a description of neither, and models fitted only on pre-break data will systematically mis-forecast the post-break series. In marketing, breaks are frequently self-inflicted and announced in advance — store openings, price restructurings, channel launches — which makes them knowable to the analyst even though they are invisible to any method that reads only the series' past.

Source: Adapted from Hyndman and Athanasopoulos (2021).

StyleCraft's series is a curriculum of breaks by design. Every store opening in the roster added a step to daily revenue: a new level, sustained, with a ramp as the store found its customers — the suburban and flagship step-ups the dataset specification plants explicitly, and the reason Section 10.3's trend panel shows stairs instead of a slope. The expansion inflection the planning lead must forecast across is exactly this: the cumulative effect of the Wave-4 openings, several recent enough that their ramps are still completing, which means the series' most recent months are governed by a regime the earlier history only partially describes. Why does history before a break mislead after it? Because every method in this chapter forecasts by assuming continuation — of a level, a trend, a rhythm — and a break is precisely a discontinuation. A model fitted through an unacknowledged opening will read the step as either an alarming trend (and over-extrapolate it) or an outlier (and ignore it); a model fitted only on pre-opening data will confidently forecast the smaller company that no longer exists. Neither model is broken. Both are answering the question they were asked — continue the past — and the past stopped being the operative regime on a date printed, in StyleCraft's case, in the stores table's opening_date column.

That last clause is the section's real payoff, and it earns the chapter's division of labor its box. Marketing's structural breaks are unusual among forecasting problems in being disproportionately scheduled: the company decides to open the store, launch the channel, restructure the prices — and writes the date down before it happens. The information exists; it simply does not live in the series. It lives in the stores table, the campaign calendar, the board deck. The analyst who joins that knowledge to the model — through the overrides of Section 10.9, or simply through choosing what history a model is allowed to learn from — is not contaminating the forecast with opinion. She is supplying the one input the method is structurally incapable of finding, and the whole verification design of Code 10.12 is built to make the point unforgettable: the model will be graded, on the known past, on an inflection the reader can see coming and the model cannot.

CONCEPT

The Calendar Is a Feature Only the Analyst Can Read

Every fitted method in this chapter consumes exactly one input: the series' own past. Everything else StyleCraft knows about its future — the openings already announced, the promotion calendar already drafted, the date Thanksgiving falls on this year, the media plan already budgeted — is invisible to the model unless the analyst carries it in. This is the forecasting form of a discipline the guide has taught since Chapter 2: the analysis serves a decision, and the decision-maker's knowledge is part of the evidence.

The working rule, enforced in the labs and the deliverable: before fitting anything, write down the known calendar — past breaks and events the model's training window contains, future breaks and events its forecast window will contain — and treat every item on the second list as an open obligation that either an override or an explicit stated assumption must discharge. A forecast delivered without its calendar is a forecast that has silently assumed nothing is scheduled to happen — which, at a company mid-expansion, six weeks before a holiday season, is not a simplification but a falsehood.

Source: Course concept developed for this guide, informed by Hyndman and Athanasopoulos (2021) and Fildes et al. (2009).

10.5 Baselines in Time: Naive and Seasonal-Naive

The toolkit proper opens where Chapter 8 taught every predictive project to open: with the opponent. Section 8.7 operationalized baselines as deliberately simple rules whose defeat is the model's entry fee, and its Table 8.3 introduced the last-value baseline — predict that the previous window repeats — noting it as the incumbent with teeth. This section is that row's time extension, and one sentence of reminder replaces any re-teach: a baseline is a fully specified prediction rule, graded on the same origins with the same metric as every candidate it opposes. What time adds is a sharper pair of rules and a humbling literature about how hard they are to beat.

The naive forecast is the last-value rule at the grain of the series: tomorrow will equal today. For a multi-day horizon it extends flat — every future day forecast at the last observed value — which sounds too simple to dignify until one remembers what it encodes: the series' most recent level, which for a trending, breaking series like StyleCraft's is often closer to the truth than an average over a history the expansion has outgrown. The seasonal-naive forecast upgrades the memory from yesterday to the rhythm: each future day is forecast by the last observed value from the same position in the seasonal cycle. Its examples must be stated at the right grain, because conflating them is how students end up believing a weekly rule carries an annual pattern. On StyleCraft's daily series at period 7, the rule is *this Saturday equals last Saturday* — and that is the only version this chapter uses. On a monthly series at period 12, the rule is *this December equals last December*. There is no clean daily version of the second: a daily annual seasonal-naive forecast would have to align a 365-day cycle against a 366-day one and would place Black Friday on the wrong day every year, which is exactly the moving-holiday problem of Section 10.4. The rule's charm, at the weekly period, is that it inherits the entire weekly pattern for free — every stable fixed-period calendar effect carried automatically, at the cost of zero fitted parameters.

DEFINITION

Naive and Seasonal-Naive Forecasts

The naive forecast predicts that every future value equals the most recent observed value — the last-value baseline of Section 8.7 applied at the series' own grain, extended flat across the horizon. The seasonal-naive forecast predicts that every future value equals the most recent observed value from the same position in the seasonal cycle at a declared period: at period 7 on daily data, each Saturday is forecast by last Saturday; at period 12 on monthly data, each December by last December. Both are fully specified, zero-parameter rules, computable from the series alone, and they are the standard opponents of time series forecasting: a fitted model that cannot beat the seasonal-naive rule on an honest evaluation has not yet demonstrated that its fitting bought anything.

Source: Adapted from Hyndman and Athanasopoulos (2021).

Students meet these rules expecting them to be strawmen, and the field's most instructive empirical tradition exists to correct the expectation. The M-competitions — decades of blind forecasting tournaments across thousands of real business series, most recently the M4 competition's one hundred thousand — return the same finding with the reliability of a season: simple methods are embarrassingly competitive, many sophisticated entrants fail to beat them, and the average gap between a tuned statistical model and a seasonal-naive rule is far smaller than the gap between either and a bad decision about which history to trust (Makridakis et al., 2020). The lesson is not that fitting is pointless — the best methods do win, and Section 10.6's family is among them — but that the win is a margin, usually a modest one, and a forecaster who cannot state her margin over seasonal-naive is a forecaster who does not know whether her model is contributing anything. StyleCraft's series is designed to teach the calibrated version of this humility: the weekly rhythm is strong, so seasonal-naive posts a respectable score, and the labs' designed expectation is that Holt-Winters beats it by a real but unspectacular margin earned almost entirely where the baseline is blind — the trend the rule cannot climb, since last Saturday is always one week of growth behind this one.

The baseline thread the guide has carried since Chapter 2 is now fully assembled, and it is worth one sentence of signposting: baselines entered as interpretive discipline (Section 2.7), operationalized into opponents (Section 8.7), and here acquire a calendar — and when Chapter 12 gives them visual form as reference lines, the thread will be complete. For this chapter's purposes the working rule is the one the frame of Code 10.6 declares before any model is fitted: the seasonal-naive forecast, at the weekly period, is the opponent; the naive forecast is reported beside it as the floor; and every exhibit in the deliverable carries the margin over both, in the declared metric, on the declared evaluation — because a forecast's value to the plan is not its accuracy but its advantage over the free rule the planning lead could have used without hiring anyone.

10.6 Smoothing: Moving Averages and the Exponential Smoothing Family

The baselines remember one value; fitted methods earn their keep by remembering more, and this section builds the chapter's working family in the order of what each layer adds. The entry instrument is the moving average: replace each day's value with the mean of a window of days around or behind it. A seven-day trailing moving average of StyleCraft's revenue erases the weekly sawtooth — each window contains exactly one of each weekday — and what remains is the smoothed level and trend, which is why the moving average is the classical decomposition's engine and the analyst's fastest reading tool for "what is the business actually doing under the rhythm." Two disciplines attach. The window is a dial: seven days removes weekly rhythm, twenty-eight removes most calendar texture, and a window wider than the feature of interest removes the feature — smoothing is deletion, and the analyst should know what she is deleting. And a moving average used for reading is centered or trailing as convenience dictates, but a moving average used inside any forecasting pipeline must be trailing only — a centered window consumes future days, which at forecast time do not exist, and Section 10.11's audit will treat a centered window in a feature the way Section 8.4 treated any information from beyond the wall.

DEFINITION

Moving Average

A moving average replaces each observation with the mean of a fixed window of surrounding observations — trailing (the window ends at the observation) or centered (the window straddles it) — smoothing away fluctuation at periods shorter than the window. A window equal to the seasonal period removes the seasonal pattern entirely, which makes the moving average both the classical instrument for extracting trend and a standing reminder that smoothing is deliberate information removal.

Source: Adapted from Hyndman and Athanasopoulos (2021).

The moving average smooths history; the exponential smoothing family turns smoothing into forecasting, and its governing idea is a graceful improvement on the flat window: weight the past by recency, with weights that decay exponentially, so that yesterday matters most and last quarter still whispers. Simple exponential smoothing maintains exactly one running quantity — the level, an exponentially weighted average of everything seen so far — updated each day by moving a fraction (the smoothing parameter) toward the newest observation. Its forecast is that level, extended flat: simple smoothing is, in effect, a naive forecast with a better memory, appropriate for series with no trend and no season, which StyleCraft's is not. Holt's method adds a second running quantity — the trend, the series' current per-day climb, itself exponentially smoothed — and forecasts by extending the level along the trend: a line, not a flat (Holt, 2004). A damped variant of Holt's method shrinks that climb as the horizon lengthens, on the empirically well-supported ground that trends estimated from recent data usually flatten rather than continuing forever; it is a candidate in this chapter's comparison precisely because a company mid-expansion invites over-extrapolation, and whether damping helps is a question for the evidence rather than for taste (Gardner, 2006). Holt-Winters completes the family by adding the third quantity this chapter has been circling: a set of seasonal effects, one per position in the cycle — seven, at StyleCraft's declared weekly period — each updated when its weekday comes around, and applied to the trended level so the forecast carries the rhythm forward (Winters, 1960). Fifty years of practice have kept the family central for the reasons this guide values: it is transparent enough to explain in a meeting, fast enough to refit weekly, and empirically hard to beat on exactly the seasonal, trending business series marketing generates (Gardner, 2006).

DEFINITION

Exponential Smoothing, Holt's Method, and Holt-Winters

Exponential smoothing is a family of forecasting methods that maintain running, recency-weighted estimates of a series' components, updating each by a smoothing parameter as new observations arrive. Simple exponential smoothing tracks a level only and forecasts it flat. Holt's method adds a smoothed trend and forecasts along it; a damped variant shrinks the projected trend as the horizon lengthens. The Holt-Winters method adds smoothed seasonal effects at a declared period and forecasts the trended level with the seasonal pattern reapplied — level, trend, and season, each with its own parameter, fitted by minimizing historical forecast error. The declared period is a modeling decision: a model fitted at period 7 carries a weekly pattern and no other.

Source: Adapted from Holt (2004), Winters (1960), and Gardner (2006).

Table 10.2 states the ladder in the form the chapter will use it: what each rung adds, what it costs, and how it fails — because each added component is also an added way to be wrong. The costs compound quietly. Each layer introduces a parameter fitted from history, and fitted parameters are flexibility in exactly Section 8.8's sense: a trend parameter can mistake a two-week surge for a climb and extrapolate it; seven seasonal effects estimated from a noisy stretch can enshrine noise as rhythm — overfitting's forecasting costume, caught the same way Chapter 8

caught it, by out-of-sample evaluation. And one failure is shared by the whole family, stated here so Table 10.2 can carry it: every rung forecasts by continuation, so no rung sees a structural break coming, and no rung at period 7 sees a holiday season coming either. The ladder climbs trends and rides weekly rhythms, and walks, at full confidence, straight past the announced opening and into December.

Table 10.2

The smoothing ladder: what each layer adds and what it costs

Method	Tracks	Adds over the rung below	Characteristic failure
Naive / seasonal-naive (Section 10.5)	Last value; last cycle at the declared period	Nothing — the free opponents	Blind to trend; one bad reference day is copied forward
Moving average	Smoothed level (a reading tool)	Noise suppression at a chosen window	Deletes features wider than intended; centered windows consume the future
Simple exponential smoothing	Level	Recency-weighted memory	Forecasts flat; trend and season leak into the level estimate
Holt	Level + trend	Climbs with the series	Extrapolates temporary surges as permanent trends
Holt, damped trend	Level + shrinking trend	Restrains long-horizon extrapolation	Under-projects genuine sustained growth; one more fitted parameter
Holt-Winters, period 7	Level + trend + weekly season	Carries the weekly rhythm forward	Overfits seasonal effects on thin stretches; carries no annual or moving calendar effect; like all rungs, cannot see a break coming

10.7 Honest Evaluation in Time: Rolling Origins, the Sealed Holdout, and Forecast Accuracy

Section 8.5 closed with a bridge this section now crosses: forecasting cannot randomly split its data, and here is why. The random split of Chapter 8 worked because its wall in time lived inside each customer's label — features before the snapshot, outcome after — so shuffling customers could not shuffle the future into the past. In a time series, the observations are the time, and a random split shatters the wall into confetti. Hold out a random twenty percent of days and every held-out day sits surrounded by training days: last Tuesday in the training set, next Tuesday in the training set, and the "test" Tuesday between them needing only interpolation — an average of its neighbors — to be predicted with flattering precision. The flattery is structural, not accidental. Adjacent days share trend, season, and even weather; a model graded on interpolation is being graded on a job that never occurs in deployment, because the deployed forecast never knows the future side of the gap. In Section 8.4's vocabulary, a random split of a time series is leakage by design: every training day after a test day is information from the future, and the resulting score measures how well the future predicts itself — Code 10.11 manufactures the demonstration, and the designed result is a shuffled "error" smaller than any honest forecast achieves.

The repair is the time-aware evaluation: put the wall back where deployment puts it. Train on the past, forecast the future, with every training observation dated strictly before every evaluated observation. The verification is a one-line date-order check — the latest training date must precede the earliest test date — and the check earns permanent membership in this chapter's audit because it is the forecasting sibling of Section 8.12's recency range check: a single violated inequality convicts an entire evaluation.

But a single contiguous holdout at the end of the series, which is where most forecasting tutorials stop, is not enough, and the reason is the same reason Chapter 8 needed cross-validation. Comparing four candidate methods on one final block makes that block part of model selection: if Holt-Winters is retained because it won there, the same block cannot also serve as an unbiased final grade. Chapter 8 solved this with k-fold cross-validation inside the training data. Time series solve it with the rolling origin.

DEFINITION

Rolling-Origin Evaluation

Rolling-origin evaluation estimates out-of-sample forecast accuracy by repeating one exercise from several successively later forecast origins: at each origin, fit on everything up to that date, forecast a fixed number of periods ahead, record the errors against what actually happened, and then move the origin forward. Every training set precedes its own test observations, so no evaluation consumes the future, and averaging across origins gives a more stable estimate than one path can. Because each origin produces one error per step ahead, the design also permits accuracy to be reported separately at each forecast horizon. It is the time series counterpart of cross-validation, and it is performed inside the training data so that a final holdout can stay sealed.

Source: Adapted from Tashman (2000) and Hyndman and Athanasopoulos (2021).

The two instruments do different jobs, and this chapter uses both in a fixed order that Table 10.3 states. Rolling origins inside the training data choose the method: naive, seasonal-naive, Holt-Winters, and damped-trend Holt-Winters are compared on the same origins with the same metrics, and the winner is named before anything else happens. Then the method is frozen and one final holdout is opened, once, for the grade. And the final holdout is chosen for a property most tutorials ignore: it must resemble the decision. This matters enough to be the chapter's largest design commitment. The plan under discussion is a holiday plan. A method graded on eight weeks of late spring has been tested on nothing that matters to it — no November–December demand, no holiday promotions, no Black Friday timing, no annual seasonal effect. So the sealed holdout in this chapter is November 1 through December 31, 2025, forecast from data ending October 31 — a prior holiday season, forecast from before it, which is the same shape as the actual commission. The result is uncomfortable and instructive: the weekly model, graded on the season it was never told about, misses by roughly twice its rolling-origin error and misses in one direction. That is the point. A holdout that resembles the decision is the only holdout that can expose the gap the decision will fall into.

Table 10.3

The two evaluation layers, and the third window that is not an evaluation at all

Layer	Window in this chapter	What it is for	What it may not do
Rolling-origin comparison	Origins every 28 days inside July 2024 – October 2025, each forecasting 56 days	Choose the method and configuration; estimate accuracy by horizon across several origins	It may not be reported as the final grade, because the candidates were compared on it
Sealed final holdout	November 1 – December 31, 2025, forecast from data ending October 31	Grade the frozen method once, on a window shaped like the decision	It may not be consulted twice, and it may not be used to pick a method
Planning forecast	July 1 – December 31, 2026, refitted on all certified history	Produce the number the plan consumes, with the calendar applied and scenarios attached	It may not be called a graded forecast; no actuals exist, and its horizon exceeds the graded one

With the where settled, the what: forecast accuracy metrics. The dollar metrics transfer whole — MAE and RMSE were built in Section 8.6 and are not re-taught; one sentence of recontextualization suffices. Each evaluated day's

error is actual revenue minus forecast revenue, MAE prices the typical daily miss in dollars, RMSE amplifies the large misses, and the choice between them still follows the decision's cost structure per Section 2.7. What forecasting adds is the metric Section 8.6 deliberately deferred to this chapter: the percentage form. Planning conversations run on percentages — "we missed the quarter by four percent" — because a percentage travels across scales: it lets the planning lead compare forecast quality across a $9,000 Tuesday and a $14,000 Saturday, across this year and last, across StyleCraft and the benchmarks in her trade association's survey. The standard instrument is MAPE, the mean absolute percentage error: each day's absolute error divided by that day's actual value, averaged, times one hundred.

DEFINITION

MAPE (Mean Absolute Percentage Error)

MAPE is the average of a forecast's absolute errors expressed as percentages of the actual values: for each period, the absolute error divided by the actual, averaged across the evaluated periods and multiplied by 100. It is scale-free, which makes it the planning world's shared currency for forecast accuracy — and it is undefined at actual values of zero and explosive near them, because a small denominator turns a modest dollar miss into an enormous percentage. MAPE is therefore reported only after its denominators are checked, and it is disqualified for series that touch or approach zero, where the dollar metrics of Section 8.6 must lead.

Source: Adapted from Hyndman and Koehler (2006).

The near-zero failure deserves the paragraph the definition promises, because it is not a curiosity — it is the most common way a competent forecast acquires a headline number that destroys its credibility. Divide by a near-zero actual and the quotient is arithmetic dynamite: Code 10.2 stages the demonstration, where a single storm-closed day of a few hundred dollars converts a forecast whose typical miss is five percent into a reported MAPE in the hundreds — one denominator, not one hundred bad forecasts, produced the number. StyleCraft's chain-level daily revenue sits comfortably above zero, so MAPE is a legitimate lead metric for this deliverable; but the moment the same pipeline is pointed at a single store's series, a single channel's, or a new store's launch weeks, near-zero actuals arrive and the metric must be benched. This chapter therefore does not leave the check to the analyst's memory. The MAPE helper of Code 10.1 asserts that every denominator is strictly positive and refuses to return a number otherwise, and the reporting helper prints the minimum denominator and a count of thin days beside every MAPE it displays. Hyndman and Koehler (2006) supply the deeper survey of scale-free alternatives for readers who meet series this guide's rule disqualifies.

One more quantity completes the evaluation vocabulary, because time gives the frame's horizon (Section 8.2) a structure it did not have at the customer grain. A forecast is not one claim but a sequence of claims — tomorrow, next week, week eight — and the claims are not equally hard. Uncertainty compounds with each step ahead: trend estimates drift further from the truth, and the world has more time to break. So expected error grows with horizon, and the plan must price it, because the December weeks it cares most about sit at the horizon's far, foggy end. But the sentence has to be said carefully, and the draft of this chapter said it too strongly. Realized error in one evaluation period need not rise monotonically, because each week ahead also differs in promotions, openings, weather, seasonal level, and plain luck — a spike in week seven may be a promotion rather than a consequence of being seven weeks out. This is exactly why horizon accuracy is estimated across several rolling origins rather than read off a single forecast path: with one path, a horizon curve is the history of one particular two months; with seven origins, each horizon has seven errors and the pattern begins to mean something. Code 10.10 builds the table that way and prints how many errors sit behind each row.

DEFINITION

Forecast Horizon

The forecast horizon is the span of future periods a forecast claims — the distance, in the series' own grain, from the last observed value to the furthest forecast one. Because uncertainty compounds with each step ahead, forecast uncertainty and expected error generally increase with horizon, although realized error need not rise monotonically in any one evaluation period. Honest evaluation therefore reports accuracy as a function of horizon rather than as a single pooled number, estimates each horizon's error from several forecast origins rather than one, and states how many errors stand behind each figure. A method's one-week-ahead grade and its eight-week-ahead grade are different facts, and the decision consumes the one that matches its own lead time.

Source: Adapted from Hyndman and Athanasopoulos (2021) and Tashman (2000).

10.8 Point Forecasts, Prediction Intervals, and Scenarios

Everything so far has graded a forecast as a number meeting a number; this section widens the forecast into what it actually is, because the widening is the deliverable's most consequential exhibit. A fitted method, asked about a future Tuesday, does not know a value — under its own assumptions it implies a distribution of plausible values, centered where the components point and spread by what the components cannot carry. The point forecast is a one-number summary of that distribution — its center, essentially — and the summary is legitimate right up until it is mistaken for the claim itself. The claim is the distribution, and the instrument that communicates it is the prediction interval.

DEFINITION

Prediction Interval

A prediction interval is a range around a point forecast derived from the forecast distribution the fitted model implies. Under the model and its assumptions, an 80% interval is constructed so that comparable forecast intervals would contain their future observations about 80% of the time. Its width is a claim about uncertainty: wide where the method knows little, growing with horizon, and honest only if the method's error assumptions are honest. A point forecast reported without an interval conceals exactly the information — the plausible range of the future — that a planning decision most needs.

Source: Adapted from Chatfield (1993) and Hyndman and Athanasopoulos (2021).

The definition's opening clause carries the qualification that keeps the instrument honest, and this chapter states it twice because it is the sentence practitioners most often drop. An interval is not a statement about the future. It is a statement about the future *under the model* — its structure, its estimated parameters, its assumption that whatever generated the past will keep generating the future. This chapter builds its intervals by simulation: draw many random future paths forward from the fitted Holt-Winters state, then read percentiles across the paths. That method propagates future disturbances through the fitted states and compounds them across the horizon, which is genuinely the largest part of the uncertainty at short horizons. What it does not do is re-estimate the model's parameters for each simulated path, so the band is conditional on the parameters that were fitted, and it prices nothing at all about the calendar the model was never told. The working statement, which belongs verbatim in the deliverable: the simulation band reflects future innovation uncertainty under the fitted Holt-Winters model and the propagation of that uncertainty across the horizon; it is conditional on the estimated parameters and may understate total uncertainty when the parameter estimates or the structural assumptions are themselves unstable. The labs make the omission

concrete rather than rhetorical: on the sealed holiday holdout the daily band achieves close to its nominal coverage while the season total lands near the top of its range, because the band prices noise faithfully and prices the missing holiday not at all.

Three readings make the interval a working tool rather than a decoration. First, width is honesty, not weakness. An interval that spans plus-or-minus twelve percent of December revenue is not the analyst hedging; it is the method reporting the size of its own ignorance, measured from its own errors — and a competitor who quotes the same future with a bare point number has not reduced the uncertainty, only concealed it. The planning instinct to prefer the confident number is precisely backward, and Section 10.15 will make the ethical version of the argument. Second, intervals fan. Because uncertainty compounds with horizon (Section 10.7), the band widens with each step ahead — the plotted forecast wears a cone, narrow tomorrow and wide in week eight — and the cone's shape is itself decision-relevant information: it tells the planning lead how much of the plan can be committed early at low risk and how much should wait for the re-forecast. It also tells her when the band has widened past usefulness, which for a period-7 method asked about a date six months out is a real and reportable finding rather than an embarrassment. Third, intervals are themselves gradable — but grading them takes more than one number.

Coverage is the obvious grade: what share of evaluated periods actually landed inside the band. It is necessary and, alone, badly incomplete, because an arbitrarily wide interval achieves perfect coverage while saying nothing. The draft of this chapter over-read a single coverage figure in both directions, treating a number near eighty as vindication and a number well above it as proof the band was too wide. Neither inference is safe on one short, serially dependent holdout, and the correction is a reporting standard rather than a rule of thumb: report the nominal coverage, the realized coverage, the average interval width, the number of evaluated periods, and coverage by horizon where the periods permit it — then read the set together. Coverage of 94 percent on sixty-one dependent daily observations is a reason to investigate, not a verdict; coverage of 70 percent is likewise a question. Distributional forecasts are ideally evaluated with instruments that reward coverage and sharpness at once, which this guide names and leaves to later coursework (Hyndman & Athanasopoulos, 2021); at this level, the discipline is simply to publish the width beside the coverage so that no one can buy the second by inflating the first.

CONCEPT

You Cannot Add Up the Edges of an Interval

The plan does not consume daily numbers. It consumes a week, a month, and a season total — so the daily interval has to be aggregated, and there is exactly one correct way to do it and one very natural wrong way. The wrong way is to add up the daily bounds: sum the seven daily 10th percentiles to get a weekly lower bound, sum the seven 90th percentiles to get an upper one. That arithmetic is not the weekly interval. The sum of seven daily 10th percentiles is the total that would occur if every one of the seven days landed at its own 10th percentile simultaneously, which is a far rarer event than the week's total landing at its 10th percentile — so the summed band is too wide, and by an amount that depends on how the days move together.

The right way is to aggregate first and quantile second. Take each simulated path, sum it over the seven days to get that path's weekly total, and then read the 10th and 90th percentiles across the paths' weekly totals. The same procedure gives the eight-week and season totals: sum each path over the whole window, then take percentiles of the totals. Dependence among the days is then handled automatically, because each path carries its own dependence with it.

How wrong is the shortcut, and in which direction? Neither question has a general answer, and overstating this is its own error. The rule is that a quantile of a sum is not the sum of the marginal quantiles: the discrepancy depends on the marginal distributions and on the dependence among the days, so the resulting band may be too wide, too narrow, or miscalibrated in ways that are not a single width at all. In StyleCraft's Holt-Winters simulations the shortcut happens to produce a wider band, because the simulated paths are strongly dependent across days — a path that drifts high stays high — so the summed edges are only a few percent too wide. Destroy that dependence by shuffling the paths day by day, as Code 10.17 does, and the same arithmetic overstates the width by a factor near the square root of seven. Same wrong formula, an order of magnitude difference in how wrong it is, and no way to know which without the paths. The correct method, by contrast, is invariant: aggregate every simulated path first, then take quantiles across the resulting totals.

Source: Course concept developed for this guide, informed by Hyndman and Athanasopoulos (2021).

The interval prices the uncertainty the method can see; the scenario prices the uncertainty it cannot. A scenario is a conditional forecast: the same machinery run under a stated assumption about the calendar the model cannot read — the openings ramp as planned, the openings stall; the holiday behaves like last year's, it behaves like the weaker of the two years on record. Scenarios and intervals answer different questions and the deliverable needs both: the interval says how wrong the method tends to be when the world stays in regime, and the scenario set says how the forecast moves when the analyst's declared assumptions do. For this plan, the practice is the three-line table the planning world already speaks — a base case carrying the declared calendar, a downside and an upside carrying the named assumption changes — with the assumptions written beside the numbers, because a scenario whose assumption is unstated is just three guesses wearing a rubric. And the deliverable must say which instrument is the planning range at which horizon. For the eight weeks after the forecast origin, the simulation band is meaningful and the scenarios refine it. For December, six months out, the band has widened past usefulness and the scenario envelope carries the range — an assumption set, honestly labeled, rather than a graded forecast wearing a graded forecast's clothes.

10.9 Judgmental Adjustment and the Documented Override

The merchant's thirty percent has been waiting since Section 10.1, and this section is where the guide keeps its promise about the one legitimate way to use it. The temptation the section disciplines is universal: every statistical forecast, presented, is immediately met by humans who know things — and some of them do. The planning lead knows the fall openings are announced; marketing knows the promotion calendar; everyone in the room knows what date Thanksgiving falls on, which is more than the model knows. A forecasting practice that ignores such knowledge wastes real information; a practice that lets every stakeholder nudge the number produces the laundered consensus Section 10.15 examines. The discipline that threads the needle is the judgmental adjustment, made under rules.

DEFINITION

Judgmental Adjustment

A judgmental adjustment is a deliberate, human-made change to a statistical forecast, intended to incorporate information the fitted method could not access — a scheduled event, a structural change, contextual knowledge outside the series. Its legitimacy depends on its provenance and its paperwork: adjustments carrying genuine event knowledge improve forecasts, while frequent small adjustments — optimism, anchoring, target pressure wearing a forecast's clothes — reliably degrade them, and the empirical record shows the difference is discipline, not intuition.

Source: Adapted from Fildes et al. (2009).

The empirical record deserves its sentence because it cuts both ways with unusual clarity. Fildes and colleagues examined more than sixty thousand forecasts and adjustments inside real supply-chain forecasting operations and found that judgmental adjustments are extremely common; that larger adjustments tended to improve accuracy more than small ones, which mostly added noise; and that positive adjustments were substantially less reliable than negative ones — upward revisions were, in effect, optimism applied with confidence (Fildes et al., 2009). The lesson is not "never touch the model" or "trust the humans" — it is that the touch must clear a bar, and the bar is information the model provably lacks. The openings on the fall calendar clear it: they are structural breaks, announced, with sizes estimable from the earlier openings' ramps — Codes 10.13 through 10.15 size one. The holiday calendar clears it: the model is fitted at period 7 and cannot carry an annual pattern, so a benchmark estimated from the prior holiday and aligned on Thanksgiving is not opinion but arithmetic the model was structurally unable to do. The merchant's gifting thesis may clear it: it is a claim about a season the history barely samples, and it can enter as a named scenario assumption with its size argued from occasionwear mix. "The team feels good about Q4" does not clear it, and the discipline exists to say so out loud.

Two implementation refinements separate an override that carries information from an override that carries a mood, and both are lessons the labs learn the hard way. The first is shape. The prose of every planning deck describes new stores as ramping — finding their customers over weeks — and then the arithmetic adds a single flat number from the first day. That is not the stated business process; it assumes the store contributes its steady-state lift immediately, and it will overshoot the first week while undershooting the fourth. An override that matches the process estimates a profile — days 1 to 7, 8 to 14, 15 to 21, 22 to 28 — from prior openings, and adds the week-specific figure. The model can stay intentionally simple; what must not stay wrong is the mismatch between what the chapter says the business does and what the code does.

The second refinement is the evidence's provenance, and it has two halves. The first is contamination control. A before-and-after mean is the natural estimator and an uncontrolled one: the four weeks after an opening differ from the four weeks before it in season, promotions, weather, neighboring openings, and organic trend, and all of that lands in the estimate alongside the store. The labs therefore control what can be controlled and disclose what cannot. Equal-length windows keep the weekday mix balanced on both sides. Openings whose windows overlap another opening are excluded, because two steps in one window cannot be separated. Any opening whose evidence window *touches* November or December is excluded — not merely those whose own date falls there, since a late-October opening measures December in its fourth week just as surely. Promotional days are counted across the pre and post windows together, because a promotion sitting in the *before* window inflates the baseline and depresses the apparent step exactly as much as one in the *after* window exaggerates it. Estimates are stratified by store type where the count permits, since a flagship and a resort store have no reason to lift equally. Every surviving opening's own estimate is printed rather than averaged away, so the reader can see the spread; the override then carries a range rather than a point, which is the only honest form for an estimate built from a handful of observations; and the code asserts that at least one uncontaminated opening survives rather than quietly averaging an empty table.

The second half is which estimate the override is allowed to use, and this is where the chapter departs from the obvious answer. A trend-tracking method fitted on a series full of recent openings has already extrapolated part of the expansion into its own trend term. Adding the raw before-and-after step on top of that adds the same growth twice. So the labs compute the lift two ways for every surviving opening: the *raw* step, which is what the series did, and the *net* step, which is the post-opening actual minus what the fitted method itself forecast for those same days — the part the model demonstrably did not already carry. The net profile is the base adjustment, because it is the only one that cannot double-count. The raw profile is retained as an upper-bound sensitivity and becomes the upside scenario of Section 10.13's envelope. The gap between the two rows is not noise to be tidied away; it is a measurement of how much of the apparent step the model was already carrying, and the override log records it. Two consequences follow that students should expect rather than be surprised by. A net week-one adjustment can legitimately be negative, which is why the log says the forecast was *adjusted* rather than *raised*. And where the net range across openings spans zero, the evidence has not established a lift at all, and the honest instrument is a named scenario rather than a point adjustment.

The instrument of the discipline is the written override, and its form is the point. Every adjustment to the statistical forecast is recorded with six elements: what changed (the periods and the amounts), the direction and size,

the evidence (the announced opening; the prior-holiday benchmark; the estimated ramp from earlier openings, with its range and its net-of-trend figure), the owner (who is accountable for this adjustment, by name), the review date (when actuals will grade it), and the baseline preserved (the unadjusted statistical forecast, kept beside the adjusted one so the adjustment's contribution can be measured after the fact). The log is the forecasting form of disciplines the guide has installed twice already — the verification log of Chapter 4, the analytic specification of Section 2.5 — and it does the same job: it converts an invisible act of judgment into an auditable decision. It also quietly settles a distinction the planning meeting will otherwise blur, and the deliverable states it in one sentence: a forecast is a probability statement about what will happen; a target is an aspiration someone chose; a plan is a commitment of resources — and the moment a target is allowed to edit the forecast that justifies the plan, all three words mean nothing, per the measures-as-targets warning of Section 2.7.

CONCEPT

The Override Is Written, or It Is Not an Override

An undocumented adjustment is indistinguishable — to the meeting, to the January review, to the analyst herself — from optimism. The written override's six elements are not bureaucracy; they are what makes human knowledge a gradable input instead of an unfalsifiable mood. The log's deepest payoff arrives at review time: with the statistical baseline preserved beside the adjusted forecast, January can compute what each override contributed — the announced-opening ramp will, on the designed data, earn its keep; the enthusiasm will not — and the organization learns, in numbers, which of its judgments deserve to keep being made. That learning loop is the entire difference between a forecasting practice that improves and one that argues.

Source: Course concept developed for this guide, informed by Fildes et al. (2009).

The section closes where the chapter's decision lives: what the forecast is for. Three planning consumers take the same forecast and stress different parts of it. The open-to-buy envelope consumes the seasonal peak and its range — receipts must be committed against a December band, and the asymmetry of Section 2.7 governs which side to lean toward, since unsold inventory becomes markdown budget while stockouts become lost revenue and disappointed stores, and the two costs are not equal. The media pacing plan consumes the weekly shape — when demand arrives, so spend can lead it — and cares more about the seasonal profile's fidelity than the level's. And targets consume the forecast last and most dangerously, as the reality check against which aspiration is set; the deliverable's job there is to keep the words separate. One forecast, three uses, three different tolerances for error — which is why the deliverable of Section 10.13 reports accuracy by horizon and shape, not as one flattering number.

10.10 Forecasting Failure Modes

The toolkit is complete; this section names the three ways it most often fails in marketing practice, each demonstrated on StyleCraft's own series and each assigned its repair. They are collected here, just before the AI section, deliberately: every one of them is a failure an assistant will commit fluently, and Table 10.4 is the checklist the audit of Section 10.11 will run.

The first and costliest is extrapolating through a structural break — the failure Sections 10.4 and 10.6 built the vocabulary for. The mechanism: a method fitted on history that ends before (or straddles) a break forecasts the regime it learned, and the regime is gone. The StyleCraft demonstration is Code 10.12: train Holt-Winters with a cutoff just before a Wave-4 opening, forecast across it, and watch a well-fitted, baseline-beating model undershoot the post-opening weeks — not because the fitting failed but because the question was malformed; the model was asked to continue a past the company had already decided to end. The repair is never more fitting; it is the analyst's calendar (Section 10.4) and the documented override (Section 10.9) — or, where a break sits inside the training window, the harder judgment of which history still describes the current regime, since a model fed pre-break years is being fed evidence about a different company.

The second is overfit seasonality — Section 8.8's disease at the calendar's grain — and this chapter's design turns it from a warning into a demonstrated boundary. The mechanism: seasonal effects are parameters, one per period position, and estimating them from thin repetitions turns noise into rhythm. StyleCraft's weekly pattern is safe: about a hundred repetitions estimate seven effects generously. Its annual pattern is the trap: two Decembers of history — one of them mid-expansion, one of them containing a store opening — are two observations of a 365-position pattern, and a pipeline coaxed into fitting granular annual seasonality will faithfully reproduce whichever accidents those two Decembers contained, including the promotion spikes and the opening ramps that were never seasonal at all. The repair is structural modesty, and this chapter enacts it rather than recommending it: weekly seasonality is fitted; annual structure is never fitted, and is carried instead by an explicit calendar layer estimated from the prior holiday and aligned on the moving date, by documented scenarios that vary that layer across the two years of evidence available, and by overrides with owners. The standing reflex is the one question that catches this everywhere: how many times has the history actually seen this pattern?

The third is the ignored promotion, the quiet one. The mechanism: promotions move revenue on chosen dates, and a pipeline that never joins the campaigns table misfiles those movements — past spikes read as seasonality (if promotions recur near the same weeks) or as noise (if they do not), and future promotions are absent entirely. The forecast then fails twice: it carries phantom lift into weeks where no promotion is planned, and it misses real lift in the weeks where one is. The StyleCraft demonstration is Code 10.5, where the ten largest decomposition remainders are joined to the campaign calendar and the hit rate is compared against the share of days that are promotional — the null expectation, printed beside the count, so that the exhibit is a comparison rather than an impression. The repair at this guide's level is the calendar discipline of Section 10.4 and the override machinery of Section 10.9: promotions declared, past ones flagged in the training window's reading and in the override's evidence windows, future ones entered as documented adjustments with sizes argued from past lifts.

Table 10.4

The three standing failure modes of marketing forecasting

Failure	Mechanism	StyleCraft demonstration	Repair
Extrapolating through a break	Methods continue the learned regime; breaks end it	Holt-Winters trained to a pre-opening cutoff undershoots the post-opening weeks (Code 10.12)	The analyst's calendar; a sized ramp override; choosing which history still applies
Overfit or absent seasonality	Seasonal effects are parameters; thin repetitions enshrine noise, and a period-7 model carries no annual pattern at all	The period-7 method misses the sealed holiday holdout by roughly twice its rolling-origin error, in one direction (Code 10.9)	Fit only well-repeated periods; carry annual and moving effects by an explicit calendar layer, benchmarks, and scenarios; count the repetitions
Ignored promotions	Chosen-date events misfiled as season or noise; future events absent	Campaign dates explain more of the largest remainders than chance predicts (Code 10.5)	Join the campaigns calendar; flag promotion days in every evidence window; enter future ones as sized, documented overrides

10.11 AI as a Forecasting Assistant

The division of labor this guide has refined across four chapters — the assistant drafts mechanics, the analyst audits meaning — meets its sternest test in forecasting, for a structural reason worth stating before the failure modes: forecasting is the first task in this guide where the standard tooling's defaults are wrong. The train_test_split habit, the shuffled cross-validation, the interpolation-friendly evaluation — the general-purpose machine learning reflexes an assistant has absorbed from a million repositories are calibrated for exchangeable rows, and time series data punishes every one of them. An assistant asked to "build a forecast and report accuracy" will, with real probability,

deliver a pipeline that shuffles days into a random split (the leakage-by-design of Section 10.7, committed in one default argument), quotes a wonderful error number earned by interpolation, and narrates the result with congratulations. The pipeline will run. The number will be excellent. And the whole artifact will be the vendor button of Section 10.1 rebuilt at home.

The forecasting-specific failure modes, named for the audit. The shuffled split: random or k-fold evaluation applied to ordered data — caught in one line by the date-order check. The reused holdout: even with a genuine time-aware block, a pipeline that compares four methods on that block and then reports the winner's score on it has spent its own grade; the repair is rolling origins for selection and one sealed block for the verdict, and an assistant will not build that separation unless the prompt demands it. The mismatched season: a holdout chosen for convenience — the last eight weeks, whatever they are — rather than for resemblance to the decision, so a holiday plan is validated on late spring. The invented season: a seasonal period guessed at 12 on daily data, or granular annual seasonality fitted from two repetitions. The silent extrapolation: a fluent forecast straight through the announced openings and straight through December, because no prompt mentioned them and no period-7 method can find them. The benched-metric revival: MAPE quoted on a series with near-zero days, the denominator check never run. The summed interval: weekly and season-total bounds produced by adding daily bounds together, which is the arithmetic error of Section 10.8 and the one most likely to survive review because the numbers look plausible. The naked point: forecasts delivered without intervals, uncertainty unmentioned. And the method upgrade: asked for a forecast, the assistant reaches for the most impressive machinery it knows — auto-tuned black-box models the student cannot explain — which fails this course's standing rule that the analyst of record must be able to explain the method to the meeting that funds it, and which usually cannot beat seasonal-naive on two years of daily retail data anyway, per Section 10.5's M-competition humility.

Where the assistant genuinely helps, use it deliberately, and in forecasting the honest list is long: the groupby-and-reconcile scaffold of Code 10.3; decomposition and plotting code, which is fiddly and perfectly specifiable; the baseline computations and the fixed leaderboard table; the rolling-origin loop itself, which is tedious date arithmetic under frozen rules — exactly what Section 8.9's discipline makes safe to delegate; error tables by horizon; the path-aggregation arithmetic of Code 10.17, once the analyst has specified which order the operations go in; and the first draft of the interval-communication paragraph, audited hard before it ships. The governing instrument extends the audit lineage one more time: Table 8.5's five points run unchanged — frame, leakage, split hygiene, baseline, error in decision units — and Table 10.5 adds the four checks forecasting makes necessary. Together they are this chapter's verification theme made procedural, and the theme's distinctive move is stated in the box: predict the number before you run the pipeline, because the analyst who has computed the baselines already knows the plausible band, and a result outside the band — in either direction — is a finding about the pipeline, not the future.

Table 10.5

The forecasting supplement to the five-point audit (run with Table 8.5)

Supplemental point	The check	Fails when
S1. Time-aware evaluation	The date-order check: latest training date strictly before earliest evaluated date, at every forecast origin; no shuffling anywhere; no feature or smoother consumes post-cutoff data	Any split call shuffles; any window is centered; train and test dates interleave
S2. Selection separated from the grade	Method comparison runs on rolling origins inside the training data; the final holdout is opened once, for the frozen method, and is chosen to resemble the decision	Candidates were compared on the block that is later reported as the grade, or the holdout season does not match the decision season
S3. Calendar and breaks declared	The known calendar written down (Section 10.4); training-window breaks and promotions identified; forecast-window events matched to overrides or stated assumptions; annual and moving effects assigned to an explicit instrument rather than to a seasonal period	The forecast extrapolates silently through an announced event or a holiday season; a period-7 model is credited with annual seasonality

Supplemental point	The check	Fails when
S4. Metric and interval honesty	Declared metric led; MAPE's denominators asserted and its minimum printed; error shown by horizon across several origins; intervals present, aggregated path-first for weekly and total figures, coverage reported beside width and period count	Metrics shuffle; MAPE hides a near-zero denominator; a point ships without its interval; weekly bounds are sums of daily bounds

AI IN PRACTICE

The Forecast You Priced Before You Ran It

The two-prompt pattern of Sections 8.11 and 9.11, with this chapter's twist: the prediction comes first, in writing, before the assistant runs anything.

Step one, the frame as specification: paste the series definition (daily revenue from the certified marketing_daily, reconciled per Code 10.3), the three windows with their dates (rolling-origin selection through October 31, 2025; sealed holdout November 1 to December 31, 2025; planning horizon July to December 2026), the declared metric (MAPE with asserted denominators and minimum printed, MAE beside it), the opponents (seasonal-naive at period 7, naive as floor), the seasonal structure to fit (weekly only; no annual fitting, under any circumstances), the known calendar (openings and promotions in every window, and the date of Thanksgiving in each year touched), and the instruction that closes the escape routes: "Compare candidates on rolling origins inside the selection window only; do not touch the sealed holdout until I name the method; report every candidate and both baselines in one table, by horizon week, with the number of errors behind each figure; produce 80% intervals and aggregate paths before quantiling for any weekly or total figure; narrate nothing yet."

Step two, the prediction: before running, write down the rolling-origin MAPE you expect for all four rows — you can compute the baselines' by hand from the series, and the model's should land between the seasonal-naive score and the designed noise floor. Then write down, separately, what you expect the sealed holiday holdout to do to that number, and why. Step three, run, then audit: Table 8.5's five points, then Table 10.5's four. Step four, only after survival: "Draft the plan paragraph: the December range under each named scenario, the margin over seasonal-naive in dollars, the overrides listed with owners and review dates." Audit its verbs — a forecast is expected, projected, likely between; it is never will be — file both exchanges per Appendix D, and sign the forecast as the analyst of record, interval and all.

Source: Course concept developed for this guide, informed by Hyndman and Athanasopoulos (2021) and Tashman (2000).

10.12 Hands-On Application in Python and Google Colab

The preceding sections built the vocabulary; this section spends it on the planning commission, in two labs that mirror the chapter's argument. Lab 10.1 makes every number touchable: the baselines and MAPE by hand at miniature scale — including the denominator failure, staged on purpose — then the certified series built, reconciled, decomposed, and audited against the campaign calendar at full scale. Lab 10.2 runs the honest evaluation: three declared windows, a rolling-origin comparison that chooses the method, a sealed holiday holdout that grades it once on a season shaped like the decision, accuracy by horizon estimated across origins, the shuffled-split demonstration, the inflection test that is this chapter's verification theme, a ramp override sized from controlled evidence and

written down, intervals aggregated in the correct order, and the actual holiday-season 2026 planning forecast with its scenarios and its declared planning unit.

The labs use the certified marketing_daily, transactions, stores, and campaigns files; column references follow the data dictionary. Three conventions hold across every cell. Every metric is computed by the helpers of Code 10.1, so no MAPE anywhere in the notebook is printed without its minimum denominator beside it. Every candidate method is a function of a training series and a horizon and nothing else, so the rolling-origin loop cannot accidentally hand one candidate information another did not get. And the notebook pins its environment. These cells are written for statsmodels 0.14.6, the current stable release, in which the simulation seed argument is spelled random_state. The development documentation for the forthcoming 0.15.0 release standardizes the argument as rng and deprecates the older name while keeping it working through a transition period, so a notebook run against that development branch will still execute and should be updated when it becomes a stable release. Pinning a stated version, which is what this chapter does, is what makes the figures reproduce exactly. The labs use statsmodels throughout (Seabold & Perktold, 2010). AI assistants may draft any code cell (Appendix C has templates; Appendix A covers Colab mechanics); every output is predicted before it is computed, and every exchange is documented per Appendix D.

10.12.1 Lab 10.1, Part A: Baselines and MAPE by Hand in Miniature

The miniature changes shape this chapter, as the data structure demands: not ten customers but fourteen days — two Monday-through-Sunday weeks of daily revenue, rounded to whole dollars for hand work and consistent with the designed scale of the certified series. Week one is the history; week two is the future to be forecast. The values carry the chapter's lessons in fourteen numbers: a weekly rhythm (quiet early weekdays, a Friday build, a weekend peak) and a gentle week-over-week climb. Before running anything, predict which rule forecasts week two better — the naive rule, which knows only yesterday, or the seasonal-naive rule, which knows the rhythm — and write one sentence of reasoning.

Code 10.1. Verify the baselines and the safe-MAPE helper

```
import pandas as pd

days = pd.date_range("2026-03-02", periods=14, freq="D")   # 2 weeks
rev = [8000, 7600, 7900, 8300, 9600, 12400, 11200,      # wk 1: history
       8400, 8100, 8200, 8900, 10100, 13000, 11900]     # wk 2: future
mini = pd.Series(rev, index=days, name="revenue")

def mae(actual, forecast):
    """Mean absolute error, in the series' own units."""
    actual = pd.Series(actual)
    forecast = pd.Series(forecast, index=actual.index)
    assert actual.notna().all() and forecast.notna().all()
    return (actual - forecast).abs().mean()

def mape(actual, forecast):
    """MAPE, with the denominator checked rather than assumed."""
    actual = pd.Series(actual)
    forecast = pd.Series(forecast, index=actual.index)
    assert actual.notna().all() and forecast.notna().all()
    assert (actual > 0).all(), (
        "MAPE is undefined at zero and unstable near it: report MAE "
        "and investigate the small denominators")
    return (actual - forecast).abs().div(actual).mean() * 100

def report(name, actual, forecast):
    """No MAPE is printed in this chapter without its denominator."""
    thin = int((actual < 0.25 * actual.median()).sum())
    print(f"{name:<24s} MAE {mae(actual, forecast):>9,.2f}"
          f"   MAPE {mape(actual, forecast):>7.2f}%"
```

```
          f"   min denom {actual.min():>9,.0f}   thin days {thin}")

week2 = mini.iloc[7:]
report("naive", week2, mini.shift(1).iloc[7:])
report("seasonal-naive", week2, mini.shift(7).iloc[7:])
```

Expected output: two lines, one per rule, each carrying MAE, MAPE, the minimum denominator behind that MAPE, and a count of thin days.

Input: fourteen literal days. Transformation: two zero-parameter forecast rules and three metrics. Output: the leaderboard of Section 10.5 at a scale where every figure can be checked by hand. The cell's real payload is the three helper functions, because they are used unchanged for the rest of the chapter and each encodes a discipline the prose argued for. mae asserts that nothing is missing before it averages. mape asserts that every denominator is strictly positive and refuses to return a number otherwise — the check of Section 10.7 made structural rather than remembered, which matters because the failure it prevents is silent. And report is the only printing route the labs use for accuracy, so the minimum denominator travels with every percentage automatically. A guide that told students to check denominators and then printed unguarded MAPEs for twenty cells would be teaching one thing and modeling another.

VERIFICATION CHECK

Before you run: compute both rules by hand. The naive rule forecasts each week-two day with the previous day's actual — its Monday forecast is Sunday's 11,200 — and its seven absolute errors (2,800; 300; 100; 700; 1,200; 2,900; 1,100) sum to 9,100, for an MAE of exactly 1,300.00; its percentage errors average to a MAPE of 12.79. The seasonal-naive rule forecasts each day with the same weekday last week, and its errors (400; 500; 300; 600; 500; 600; 700) sum to 3,600, for an MAE of 514.29 and a MAPE of 5.25. Predict also what the minimum-denominator column will show, and why it will reassure you here and alarm you in Code 10.2.

After you run: reconcile every figure, then read the anatomy of the defeat, because it is Table 10.2's first row demonstrated on fourteen numbers. The naive rule is not uniformly bad — midweek, where tomorrow does resemble today, its misses are tiny — but it is blindsided at every turn of the rhythm, forecasting Monday from Sunday's peak and Saturday from Friday's build, and the rhythm turns twice a week. The seasonal-naive rule inherits the rhythm free and loses only to the week-over-week climb: all seven of its errors are positive, a persistent undershoot of roughly five percent — bias, not noise — which is precisely the signature of a trend the rule has no way to carry, and precisely the margin a trend-tracking method exists to claim. Write the two sentences the miniature has earned: the weekly rhythm is worth about 786 dollars of daily MAE (1,300.00 against 514.29), and whatever beats seasonal-naive must earn its living from the climb.

Investigate if: your hand arithmetic disagrees with the printed figures anywhere. Every number in this cell is reproducible with a calculator, and a mismatch is arithmetic rather than convention.

The second half of the miniature stages the metric failure of Section 10.7. Suppose a storm closes the stores on week two's Tuesday and revenue collapses to 400 dollars. Rerun the seasonal-naive evaluation against the storm-struck actuals — predict both metrics' movements first.

Code 10.2. Demonstrate MAPE's denominator failure

```
snaive_fc = mini.shift(7).iloc[7:]
storm = week2.copy()
storm.iloc[1] = 400                       # the storm-closed Tuesday

report("seasonal-naive", week2, snaive_fc)
```

```
report("seasonal-naive, storm", storm, snaive_fc)

ape = (storm - snaive_fc).abs().div(storm) * 100
worst = ape.idxmax()
print(f"\nthe one term that produced the headline: {worst.date()}"
      f"  actual {storm[worst]:,.0f}"
      f"  forecast {snaive_fc[worst]:,.0f}  APE {ape[worst]:,.0f}%")
print(f"days forecast within 10%: {int((ape <= 10).sum())} of {len(ape)}")
```

Expected output: the clean and storm-struck seasonal-naive rows side by side, then the single date responsible for the headline with its absolute percentage error, and a count of days that were forecast within ten percent.

Input: the same fourteen days with one value replaced. Transformation: nothing but the substitution. Output: the most instructive pair of numbers in the chapter. The MAE moves honestly, from 514.29 to 1,471.43, because the forecast genuinely missed a collapsed day by a lot. The MAPE does something else: the Tuesday term is 7,200 divided by 400, or 1,800 percent, one term of a seven-term average, and the reported figure explodes from 5.25 to 261.52. Verify the arithmetic by hand — the sum of the seven percentage errors divided by seven — and then note what the guarded helper did and did not do. It did not refuse: 400 is positive, so the assertion passes and the number is computed. What it did was print the minimum denominator and the thin-day count beside the number, which is the difference between a metric that lies and a metric that lies while showing you why.

VERIFICATION CHECK

Before you run: predict both metrics' movements in direction and rough size. Then predict the thin-day count, and state what value of the storm Tuesday would have made the mape helper stop rather than compute.

After you run: write the sentence that must accompany every MAPE this pipeline ever reports — the denominator, not the forecast, produced the number. Six of the seven days were forecast within ten percent; the headline says the method missed by two hundred sixty. Then write the honest one-line summary of the storm week that the standing rule requires: the MAE moved from 514 to 1,471, one day closed, and MAPE is disqualified for this week.

Investigate if: you are tempted to fix this by dropping the storm day. Say out loud what that would mean — deleting the largest genuine forecast miss in the window because it embarrasses a chosen metric — and then reach for the metric that survives instead of editing the data that does not flatter it.

10.12.2 Lab 10.1, Part B: The Daily Series, Reconciled, Decomposed, and Audited

Part B manufactures the real object. The certified marketing_daily ships the daily series; the known-truth discipline requires rebuilding its revenue column from the fact table before trusting it — the grain change of Section 3.3, executed with Chapter 4's groupby, cited and not re-taught. Two mechanical traps sit in the reconciliation and the draft of this chapter fell into both, so read the cell for them specifically.

Code 10.3. Build and reconcile the daily series

```
import numpy as np

md = pd.read_csv("marketing_daily.csv", parse_dates=["date"])
md = md.set_index("date").sort_index().asfreq("D")

# asfreq inserts missing calendar days as NaN; it does not make them
# zero-revenue days. A quiet day is a zero, not a hole (Section 10.2) --
# so fill it explicitly and state the rule. Where a gap instead means an
# incomplete extract, stop rather than filling silently.
gaps = md["revenue"].isna()
```

```
print("calendar days with no source row:", int(gaps.sum()))
assert gaps.sum() <= 3, "too many gaps to be closures -- audit it"
md.loc[gaps, "revenue"] = 0.0

tx = pd.read_csv("transactions.csv", parse_dates=["order_date"])
daily_tx = (tx.groupby("order_date")["line_revenue"].sum()
              .reindex(md.index, fill_value=0.0)   # quiet day = 0
              .rename("revenue_from_tx"))

assert md.index.is_monotonic_increasing and md.index.is_unique
assert md["revenue"].notna().all()
assert np.isclose(md["revenue"], daily_tx, atol=0.01).all(), (
    "the shipped series disagrees with the transaction roll-up")

series = md["revenue"].rename("revenue")
print("days:", len(series), " span:", series.index.min().date(),
      "->", series.index.max().date())
print(f"zero-revenue days: {int((series == 0).sum())}"
      f"   minimum: {series.min():,.0f}"
      f"   median: {series.median():,.0f}")
```

Expected output: the count of calendar days with no source row, then the series length, span, zero-revenue count, minimum, and median; three assertions pass silently.

Input: the certified daily table and the transaction fact table. Transformation: a complete-calendar reindex, an explicit zero-fill rule, a groupby roll-up reindexed onto the same calendar, and an element-wise comparison. Output: series, the certified daily object every later cell uses, plus the evidence that it means what Section 10.2 says it means.

The two traps are worth naming because they are silent. First, asfreq builds a complete daily index and inserts the missing days as blanks, not as zeros — so a series that "has no gaps" after an asfreq call may be full of blanks that every later arithmetic operation quietly propagates or skips. The rule from Section 10.2 has to be executed: a day with no transactions is a zero, so fill it and say so; a run of blanks too long to be closures is an incomplete extract, so stop. The cell does both, with the threshold visible in the assertion rather than buried in a comment. Second, the transaction roll-up omits dates on which nothing was sold, so comparing it to the daily table by index alignment compares two differently shaped objects and can report agreement on the intersection while ignoring exactly the days in dispute. Reindexing the roll-up onto the series' own calendar with fill_value=0.0 forces both sides onto the same days before anything is compared, and the closeness check then runs on every day in the series rather than on whichever days both objects happened to contain.

VERIFICATION CHECK

Before you run: write three predictions. The span: the certified history runs July 2024 through June 2026, so predict roughly 730 rows. The reconciliation: the closeness assertion should pass on every day; anything less is the lab's first exercise in aggregation forensics — find one disagreeing date and determine which side dropped or double-counted a line, per the join disciplines of Chapter 4. The zero count: predict it near zero for a chain with an always-on online channel, and write down what a nonzero count would oblige you to investigate.

After you run: confirm the assertions passed and record the minimum daily revenue in your notes, because it is the number that licenses MAPE for the rest of the chapter. Then state, in one sentence, what would change about the whole chapter if the minimum were 40 dollars rather than several thousand.

Investigate if: the gap count is above zero. Before filling anything, look at the dates: a scatter of isolated days is plausibly closures, a contiguous run at either end of the file is a truncated extract, and the two require opposite responses.

Code 10.4. Decompose the series and mark the openings

```
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

RESERVED = {"S00", "S99"}        # Online and App are not physical stores
stores = pd.read_csv("stores.csv", parse_dates=["opening_date"])
opens = (stores[~stores["store_id"].isin(RESERVED)]
         .dropna(subset=["opening_date"])
         .sort_values("opening_date"))
in_window = opens[opens["opening_date"]
                  .between(series.index.min(), series.index.max())]

dec = seasonal_decompose(series, model="additive", period=7)

fig, ax = plt.subplots(4, 1, figsize=(9, 9), sharex=True)
panels = [(series, "observed"), (dec.trend, "trend"),
          (dec.seasonal, "seasonal, period 7"), (dec.resid, "remainder")]
for a, (part, label) in zip(ax, panels):
    a.plot(part.index, part.to_numpy(), lw=0.8)
    a.set_ylabel(label, fontsize=9)
for d in in_window["opening_date"]:      # the calendar HW cannot read
    for a in ax[:2]:
        a.axvline(d, color="#57068C", lw=0.8, ls="--")
ax[1].set_title("dashed: store openings from the stores table",
                fontsize=9, loc="left")
fig.tight_layout()

print("openings inside the series window:", len(in_window))
print(in_window[["store_name", "store_type", "opening_date"]]
      .to_string(index=False))
swing = dec.seasonal.max() - dec.seasonal.min()
print(f"\nweekly peak-to-trough: {swing:,.0f}"
      f"  ({swing / series.mean():.0%} of mean daily revenue)"
      f"  <- what seasonal-naive carries free")
```

Expected output: a four-panel decomposition figure with the store-opening dates drawn as dashed vertical lines on the observed and trend panels, then the count and roster of openings inside the series window, then the weekly seasonal swing in dollars and as a share of mean daily revenue.

Input: the certified series and the stores roster. Transformation: an additive decomposition at period 7, and a filter that removes the reserved non-physical locations before anything is plotted. Output: the reading instrument of Section 10.3, with the calendar the model cannot read drawn on top of it. Two details are deliberate. The reserved set is removed by store identifier rather than by hoping that Online and App carry a null opening date, because a roster row that acquires a date later would otherwise silently enter every opening analysis in the chapter. And the openings are marked on the plot rather than described in a caption, because the whole argument of Section 10.4 is that the steps land where the calendar says they do, and a reader who has to take that on trust has not seen the evidence.

The seasonal swing printed at the end is the number Section 10.5 promised: the peak-to-trough height of the weekly pattern is, to a first approximation, exactly what the seasonal-naive rule carries for free and what any fitted model must therefore stop congratulating itself for.

VERIFICATION CHECK

Before you run: sketch what you expect from the designed story — an upward drift, visible stair-steps at the opening dates, two November–December surges, a late-August lift, and a weekly sawtooth over all of it. Predict where in the four panels each of those five features will appear, and name the one that will appear in the remainder panel because a period-7 decomposition has nowhere else to put it.

After you run: write one sentence per panel. Confirm that the trend panel's steps line up with the dashed opening lines rather than merely resembling them. Confirm that the seasonal panel is a stable seven-day sawtooth across both years, and record its dollar height. Then look hard at the remainder panel around each November and December, and say what the presence of large remainders there proves about what a period-7 seasonal component does and does not carry — this is the visual form of the boundary Section 10.4 declared and Code 10.9 will grade.

Investigate if: the trend panel's steps do not align with the opening dates, or if the seasonal panel drifts in amplitude across the two years. The first suggests a roster date that does not match the series; the second suggests the additive assumption is straining and a multiplicative decomposition is worth reading beside it.

Code 10.5. Audit the remainder against the campaign calendar

```
campaigns = pd.read_csv("campaigns.csv",
                        parse_dates=["start_date", "end_date"])
promo_days = pd.DatetimeIndex(np.unique(np.concatenate(
    [pd.date_range(r.start_date, r.end_date, freq="D").to_numpy()
     for r in campaigns.itertuples()]))).intersection(series.index)

top10 = dec.resid.abs().nlargest(10).index.sort_values()
audit = pd.DataFrame({
    "date": top10.date,
    "revenue": series.loc[top10].round(0).to_numpy(),
    "remainder": dec.resid.loc[top10].round(0).to_numpy(),
    "in_promo": top10.isin(promo_days)})
audit["campaign"] = [
    ", ".join(campaigns.loc[(campaigns["start_date"] <= d)
                            & (campaigns["end_date"] >= d),
                            "campaign_name"].unique()) or "--"
    for d in top10]
print(audit.to_string(index=False))

hits, share = int(audit["in_promo"].sum()), len(promo_days) / len(series)
print(f"\n{hits} of the 10 largest remainders sit in a campaign window")
print(f"promotion days are {share:.0%} of the series, so chance alone "
      f"predicts about {10 * share:.1f} of 10")
print("unexplained days, for the override watch list:",
      [str(d) for d in audit.loc[~audit["in_promo"], "date"]])
```

Expected output: a ten-row table of the largest absolute remainders with revenue, remainder, a promotion flag, and the campaign names covering that date; then the promotion hit count, the share of all days that are promotional as the chance expectation, and the list of unexplained dates.

Input: the decomposition remainder and the campaigns table. Transformation: a date-expanded promotion calendar, a top-ten selection by absolute remainder, and a join. Output: Section 10.10's third failure mode caught before any forecast has been run, and the override log's opening watch list.

The line that makes this an exhibit rather than an impression is the chance expectation. If a fifth of all days sit inside some campaign window, then two of any ten days will land inside one by accident, and a hit count of three proves nothing. Printing the null beside the count is the difference between "promotions explain the spikes" as a finding and as a feeling — the same discipline Section 10.5 applies to model accuracy, applied here to a pattern claim. The unexplained dates are then genuinely interesting: they are the days the decomposition could not explain and the campaign calendar cannot either, and on StyleCraft's designed data they cluster where you would expect once you have read Code 10.4's remainder panel.

10.12.3 Lab 10.2, Part A: Three Windows and the Opponents

Lab 10.2 opens, per the standing rule, with the frame in a markdown cell — the predictive frame of Section 8.2 wearing this chapter's clothes. Unit of prediction: one prediction per future day, rolled up to declared weekly blocks for the deliverable. Target: daily chain revenue as reconciled in Code 10.3. Information: the series' own past up to each forecast origin, plus the declared calendar (openings from the stores table; promotions from the campaigns table; the date of Thanksgiving in each year the analysis touches). Declared metric: MAPE with asserted denominators and minimum printed, MAE beside it, error shown by horizon week. Opponents: seasonal-naive at period 7, naive as floor. Seasonal structure to fit: weekly only. And then the part that is new to this chapter and is the whole point of Table 10.3 — not one holdout but three windows, with different rights.

Code 10.6. Declare the three evaluation windows

```
# THE FRAME IN CODE. Three windows, declared before anything is fitted.
SERIES_END    = series.index.max()                       # 2026-06-30
SELECT_END    = pd.Timestamp("2025-10-31")   # rolling-origin selection
HOLDOUT_START = pd.Timestamp("2025-11-01")   # sealed holiday holdout
HOLDOUT_END   = pd.Timestamp("2025-12-31")
PLAN_START    = pd.Timestamp("2026-07-01")   # the actual planning horizon
PLAN_END      = pd.Timestamp("2026-12-31")

H = 56              # rolling-origin forecast length, in days
MIN_TRAIN = 240     # shortest training window any candidate may see
ORIGIN_STEP = 28    # days between successive forecast origins
MAE_TOLERANCE = 0.05   # the selection rule's practical-tie band

select = series.loc[:SELECT_END]
holdout = series.loc[HOLDOUT_START:HOLDOUT_END]

assert select.index.max() < holdout.index.min()   # date-order check S1
assert holdout.index.min() == HOLDOUT_START
assert holdout.index.max() == HOLDOUT_END
assert PLAN_START == SERIES_END + pd.Timedelta(days=1)

origins = pd.date_range(select.index.min() + pd.Timedelta(days=MIN_TRAIN),
                        SELECT_END - pd.Timedelta(days=H),
                        freq=f"{ORIGIN_STEP}D")
assert len(origins) >= 5, "too few origins to average over"

print(f"selection : {select.index.min().date()} -> {SELECT_END.date()}"
      f"   {len(select)} days, {len(origins)} origins, H={H}")
print(f"holdout   : {HOLDOUT_START.date()} -> {HOLDOUT_END.date()}"
      f"   {len(holdout)} days -- a holiday season, like the decision")
print(f"plan      : {PLAN_START.date()} -> {PLAN_END.date()}"
      f"   {(PLAN_END - PLAN_START).days + 1} days, no actuals exist")
print("origins:", ", ".join(str(o.date()) for o in origins))
```

Expected output: three lines describing the selection, holdout, and planning windows with their lengths, then the list of forecast origins; four assertions pass silently.

Input: the certified series. Transformation: date declarations and an origin schedule. Output: the objects every later cell in Lab 10.2 consumes, and the assertions that make the design auditable in the notebook rather than asserted in a methods paragraph. The date-order assertion is supplement point S1 made executable, and the lab requires it in every notebook this chapter grades. The assertion that the planning horizon begins the day after the series ends is the smaller and more easily broken one: it is what stops a "future" forecast from silently overlapping history that already exists, which is the version of leakage that survives every other check because the dates look tidy.

Read the three windows against Table 10.3 and say what each may and may not do. The selection window may be consulted as often as the analyst likes, because it is training data. The holdout may be consulted once, for one frozen method, and its date range was chosen for a reason no convenience-based holdout can claim: it is a holiday season, forecast from before it, which is the shape of the decision. The planning window may not be graded at all, because no actuals exist — and the honest label on it is not "forecast" but "forecast plus declared calendar plus stated assumptions," which is what Sections 10.9 and 10.13 spend their time enforcing.

Code 10.7. Declare the candidates, the model factory, and the selection rule

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# ONE place a model is fitted. Every later cell that needs a fitted
# object calls this with CHOSEN, so the method that wins the comparison
# is the method that produces the intervals and the planning forecast.
MODEL_CONFIG = {"Holt-Winters": False, "Holt-Winters, damped": True}


def fit_candidate(name, train):
    """None for the zero-parameter baselines, which fit nothing."""
    if name not in MODEL_CONFIG:
        return None
    return ExponentialSmoothing(
        train, trend="add", damped_trend=MODEL_CONFIG[name],
        seasonal="add", seasonal_periods=7,
        initialization_method="estimated").fit()


def fit_forecast(name, train, h):
    """Every candidate is a function of (train, h) and nothing else."""
    if name == "naive":
        return np.repeat(float(train.iloc[-1]), h)
    if name == "seasonal-naive":
        last = train.iloc[-7:].to_numpy()
        return np.tile(last, int(np.ceil(h / 7)))[:h]
    return fit_candidate(name, train).forecast(h).to_numpy()


CANDIDATES = ["naive", "seasonal-naive", "Holt-Winters",
              "Holt-Winters, damped"]
SIMPLICITY = {n: i for i, n in enumerate(CANDIDATES)}   # tiebreak order
print(f"candidates: {len(CANDIDATES)}   practical-tie band: "
      f"{MAE_TOLERANCE:.0%} of the best rolling MAE")
print("""selection rule, declared before any fit: among the candidates
inside the tie band on rolling MAE, take the lowest week-8 MAE, then
the smallest absolute bias, then the simpler method.""")
```

Expected output: the candidate count, the declared tie band, and the selection rule printed in full; nothing is fitted here.

Input: none. Transformation: two declarations and two function definitions. Output: the single place in the chapter where a model is fitted, and the rule that will name the winner — both written before any evidence exists.

This cell exists because of a defect the first revision of this chapter still carried, and the defect is worth naming because it is the most common way a careful forecasting notebook quietly cheats. The comparison selected a method; the interval and the planning forecast were then built by a separately typed Holt-Winters call. Whenever the comparison happened to choose something else — the damped variant, or a baseline — the deliverable silently came from a method that had never won anything. The repair is structural: fit_candidate is the only route to a fitted object, every later cell calls it with CHOSEN, and the baselines return None because they genuinely fit nothing, which forces the interval cell to declare what it will do when a zero-parameter rule wins rather than substituting a convenient model that can simulate.

The selection rule is the second declaration, and it is precommitted for exactly the reason Section 8.9 gave: a rule written after the leaderboard is visible can be written to produce the preferred winner. The rule states a practical-tie band on rolling MAE, then breaks ties on the two columns the prose says the plan cares about — the week-eight grade the December weeks will consume, and the absolute bias that says whether a method is systematically low — and prefers the simpler method last. Note what the band buys: a candidate that wins the pooled average by a fraction of a percent no longer wins automatically, which is the numeric form of the chapter's own warning that a margin smaller than the noise is not yet a finding.

Code 10.8. Run the rolling-origin comparison and apply the rule

```
rows = []
for name in CANDIDATES:
    e_all, ape_all, wk8, signed = [], [], [], []
    for origin in origins:
        train = series.loc[:origin]
        test = series.loc[origin + pd.Timedelta(days=1):][:H]
        e = test.to_numpy() - fit_forecast(name, train, H)
        e_all.append(np.abs(e))
        ape_all.append(np.abs(e) / test.to_numpy() * 100)
        wk8.append(np.abs(e)[-7:])
        signed.append(e)
    rows.append({"candidate": name,
                 "rolling MAE": np.concatenate(e_all).mean(),
                 "MAPE %": np.concatenate(ape_all).mean(),
                 "week-8 MAE": np.concatenate(wk8).mean(),
                 "bias": np.concatenate(signed).mean()})

board = pd.DataFrame(rows).set_index("candidate").round(2)
print(f"{len(origins)} origins x {H} days = {len(origins) * H} "
      "daily errors per candidate\n")
print(board.to_string())

best = board["rolling MAE"].min()
band = best * (1 + MAE_TOLERANCE)
eligible = board[board["rolling MAE"] <= band].copy()
eligible["abs bias"] = eligible["bias"].abs()
eligible["simplicity"] = [SIMPLICITY[i] for i in eligible.index]
CHOSEN = eligible.sort_values(
    ["week-8 MAE", "abs bias", "simplicity"]).index[0]

col = board["rolling MAE"]
print(f"\nwithin the {MAE_TOLERANCE:.0%} tie band: {len(eligible)} of "
      f"{len(board)} candidates -> {list(eligible.index)}")
win = board.loc[CHOSEN]
print(f"SELECTED: {CHOSEN}   week-8 MAE {win['week-8 MAE']:,.0f}"
      f"   bias {win['bias']:+,.0f}")
print(f"margin over the opponent: "
      f"{col.loc['seasonal-naive'] - col.loc[CHOSEN]:,.0f} of daily MAE."
      "\nThe sealed holdout is still closed.")
```

Expected output: the number of daily errors behind each candidate, a four-row table carrying rolling-origin MAE, MAPE, week-eight MAE and mean signed error for naive, seasonal-naive, Holt-Winters and damped-trend Holt-Winters, then the selected method, its margin over the opponent in daily MAE, and a statement that the sealed holdout is still closed.

Input: the selection window only. Transformation: for each candidate and each origin, fit on everything up to the origin and forecast the next fifty-six days, then pool the errors. Output: the leaderboard that names the method — and the only evidence the analyst is permitted to use in naming it.

Two design choices carry the section's arguments. Every candidate is expressed as one function of a training series and a horizon, including the two baselines, so that "same origins, same metric, same baselines" is enforced by the code rather than remembered by the analyst — the direct time-series analogue of Chapter 8's insistence that a baseline be an estimator scored by the same call. And the table carries four columns rather than one, because a single pooled MAE hides the two things the plan cares about: the week-eight column is the far-horizon grade the December weeks will actually consume, and the bias column is the signed error, which is where a trending, expanding series announces whether a method is systematically low. Those two columns are not decoration — the rule declared in Code 10.7 sorts on them, so a candidate that wins the pooled average inside the tie band but carries a worse far-horizon grade does not win the chapter.

VERIFICATION CHECK

Before you run: predict the ordering of all four candidates on rolling-origin MAE, and predict the sign of the bias column for each. Then predict how many candidates will fall inside the declared tie band, and say what you will conclude if the answer is more than one. Write down the band you would consider implausible in either direction — far worse means a broken fit, implausibly better means the evaluation is dirty.

After you run: grade every prediction. Confirm the arithmetic behind the error count: origins times horizon length, which is the number this table's stability rests on. Then read the bias column across all four rows and write the sentence it forces — every candidate that cannot climb is biased low on an expanding series, which is a fact about the business, not about the software. State the winner's margin in dollars per day and in dollars across a fifty-six-day window, because the second is the sentence that survives into the deliverable.

Investigate if: the damped variant wins, or if naive beats seasonal-naive. The designed expectation is that the undamped Holt-Winters wins on this data, but a different result is a finding to trace, not an error to edit away. A damped winner says the series' recent trend is not projecting well and the plan should expect flatter growth than the raw trend implies — and it is precisely the case the model factory of Code 10.7 exists to carry safely into the interval and the planning forecast. Naive beating seasonal-naive would say the weekly rhythm has broken down, which on this data would mean the series or the reconciliation is wrong, and Code 10.3 is where you go. If a baseline wins outright, read the interval cell's fallback branch before doing anything else.

10.12.4 Lab 10.2, Part B: The Sealed Holiday Holdout

The method is now frozen. Part B opens the sealed window once, and it does so with one addition declared before the window opens: the calendar layer. Section 10.4 established that a period-7 model carries no annual pattern, and the honest response is not to hope but to supply one. The layer is estimated entirely inside the training data by running the same evaluation a year earlier — fit through October 2024, forecast the 2024 holiday, and record the ratio of what actually happened to what the method said, day by day. That ratio is what the weekly model misses in a holiday, measured rather than assumed. And it is indexed by days from Thanksgiving rather than by calendar date,

because Thanksgiving moved between the two years and a day-of-year index would misalign the Black Friday weekend by a full day in one direction.

Code 10.9. Open the sealed holiday holdout, once

```
def thanksgiving(year):
    """Fourth Thursday in November -- a date the calendar moves."""
    first = pd.Timestamp(f"{year}-11-01")
    first += pd.Timedelta(days=(3 - first.dayofweek) % 7)
    return first + pd.Timedelta(days=21)

def holiday_index(s, year, method):
    """Actual over statistical forecast, day by day, across one prior
    holiday -- indexed by DAYS FROM THANKSGIVING so that the moving
    retail events line up instead of drifting by weekday."""
    cut = pd.Timestamp(f"{year}-10-31")
    actual = s.loc[f"{year}-11-01":f"{year}-12-31"]
    fc = pd.Series(fit_forecast(method, s.loc[:cut], len(actual)),
                   index=actual.index)
    idx = (actual / fc).rolling(7, center=True, min_periods=4).mean()
    idx.index = (actual.index - thanksgiving(year)).days
    return idx

IDX24 = holiday_index(select, 2024, CHOSEN)
print("Thanksgiving:", ", ".join(f"{y} {thanksgiving(y).date()}"
                                 for y in (2024, 2025, 2026)))
print(f"2024 index: min {IDX24.min():.2f}  peak {IDX24.max():.2f}"
      f" at day {IDX24.idxmax():+d}")

train, h = select, len(holdout)
fc_stat = pd.Series(fit_forecast(CHOSEN, train, h), index=holdout.index)
sn_fc = pd.Series(fit_forecast("seasonal-naive", train, h),
                  index=holdout.index)
off25 = pd.Series((holdout.index - thanksgiving(2025)).days,
                  index=holdout.index)
fc_cal = fc_stat * off25.map(IDX24).ffill()

print("\n--- SEALED HOLDOUT OPENED ONCE: Nov 1 - Dec 31, 2025 ---")
for name, fc in [("seasonal-naive", sn_fc), (CHOSEN, fc_stat),
                 (f"{CHOSEN} + holiday calendar", fc_cal)]:
    report(name, holdout, fc)
    bias = (holdout - fc).mean()
    print(f"{'':<24s} signed error {bias:>+9,.0f}"
          f"   season total miss "
          f"{(fc.sum() - holdout.sum()) / holdout.sum():>+7.1%}")
```

Expected output: the Thanksgiving dates for 2024, 2025 and 2026; the 2024 holiday index's minimum, peak, and the offset from Thanksgiving at which the peak falls; then three graded rows — seasonal-naive, the frozen method, and the frozen method with the holiday calendar — each with MAE, MAPE, minimum denominator, mean signed error, and the season-total miss as a percentage.

Input: the selection window and the sealed holdout. Transformation: one calendar layer estimated from the prior holiday, and three forecasts scored on the sealed window. Output: the chapter's central grade, and the exhibit that proves the boundary of Section 10.4 rather than asserting it.

Read the three rows in order and the design explains itself. Seasonal-naive repeats late October forever and is destroyed by the season. The frozen Holt-Winters, which beat every candidate on rolling origins, misses the holiday by roughly twice its rolling-origin error and misses it in one direction — a large positive signed error, which is to say a systematic undershoot, which is to say exactly what a model that carries a weekly rhythm and no annual pattern must do when December arrives. Adding the calendar layer cuts the miss substantially and, on this data,

overshoots slightly in the other direction, because the single prior holiday available for estimating it contained a store opening of its own and therefore overstates the uplift a little. Every part of that sentence belongs in the deliverable. An override estimated from one observation is an assumption with a range, and the direction of its error on the one occasion it could be graded is the most useful thing anyone knows about it.

VERIFICATION CHECK

Before you run: this is the prediction the chapter has been building toward, so write it in full sentences. Predict the sign of the signed-error column for seasonal-naive and for the frozen method, and say why both signs are forced rather than likely. Predict how the sealed-holdout MAPE compares to the rolling-origin MAPE of Code 10.8, and by roughly what factor. Then predict whether the calendar layer will over- or under-correct, and name the contaminating event in the 2024 holiday window that decides the answer — the roster from Code 10.4 has it.

After you run: confirm the signs, then write the two sentences the deliverable needs. First: the method chosen by rolling origins on non-holiday windows is graded, on the season the plan is actually about, at a materially worse error than its selection score — so the selection score may not be quoted as the plan's expected accuracy. Second: the gap between those two numbers is not a defect in the model but the measured size of the calendar the model was never given, which is why the analyst's layer is a component of the forecast rather than a garnish on it. Then record the calendar layer's own direction of error, with its cause, in the override log's evidence field.

Investigate if: the calendar layer makes things worse rather than better, or if the frozen method's signed error is negative. The designed expectation is a large positive signed error for both baseline and model, and a calendar layer that reduces the miss; a different result is a finding to trace, not an error to edit away. A layer that makes things worse would say the prior holiday is too contaminated to serve as a benchmark, and the response is a scenario range rather than a point adjustment. A negative signed error would mean the model is over-forecasting the holiday, which on a period-7 fit would be surprising enough to send you back to the trend term and to Code 10.4's trend panel.

Code 10.10. Measure accuracy by horizon across origins

```
rows = []
for origin in origins:
    train = series.loc[:origin]
    test = series.loc[origin + pd.Timedelta(days=1):][:H]
    e = np.abs(test.to_numpy() - fit_forecast(CHOSEN, train, H))
    ape = e / test.to_numpy() * 100
    block = np.arange(H) // 7 + 1        # eight blocks of exactly 7 days
    for w in range(1, H // 7 + 1):
        m = block == w
        rows.append({"week_ahead": w, "MAE": e[m].mean(),
                     "MAPE": ape[m].mean()})

by_h = (pd.DataFrame(rows).groupby("week_ahead")
        .agg(origins=("MAE", "size"), MAE=("MAE", "mean"),
             MAPE=("MAPE", "mean"), MAE_sd=("MAE", "std")).round(2))
by_h["daily_errors"] = by_h["origins"] * 7      # 7 days per weekly block
print(by_h.to_string())
rise = by_h["MAE"].iloc[-1] / by_h["MAE"].iloc[0]
print(f"\nweek 8 MAE is {rise:.2f}x week 1's")
print("monotone across all eight weeks:",
```

```
      bool((by_h["MAE"].diff().dropna() > 0).all()),
      "  <- expected error rises; realized error need not, every step")
```

Expected output: an eight-row table giving, per week ahead, the number of forecast origins behind the figure, the mean MAE and MAPE, the standard deviation of MAE across origins, and the implied count of daily errors; then the week-eight to week-one ratio and a flag stating whether the increase was monotone.

Input: the selection window's origins. Transformation: fifty-six-day forecasts from each origin, cut into eight consecutive seven-day blocks, averaged by block across origins. Output: the horizon curve of Section 10.7, estimated the way it has to be estimated.

Three properties of this table are corrections to the draft of this chapter, and each is worth reading for. The blocks are exactly seven days counted from the forecast origin, not calendar weeks from a resample call, so no block is a partial week and no bin boundary is decided by a library default. Two count columns appear rather than one, and the distinction matters: origins is the number of independent forecast paths behind the row, and daily_errors is that number times seven, because each origin contributes a whole week of daily errors to its weekly MAE. Quoting the larger figure as if it were the number of independent observations would overstate the row's stability sevenfold; quoting only the smaller one would understate how much data the mean rests on. The table prints both. And the monotone flag is printed rather than assumed, because the honest claim is that expected error and uncertainty grow with horizon while realized error in a finite sample need not rise at every step. On StyleCraft's data it does not: some middle week will sit below the week before it, and the standard-deviation column shows why — the spread across origins at a given horizon is comparable to the differences between adjacent horizons.

VERIFICATION CHECK

Before you run: predict the direction of the MAE column and, separately, predict whether it will rise at every single step. Predict the week-eight to week-one ratio to within a factor of two. Then write down what you will conclude if a middle week dips.

After you run: grade both predictions and write the sentence the plan needs — the December-distance weeks carry roughly the week-seven and week-eight grade, not the headline average. Then find the calendar event sitting under any week that spikes or dips out of pattern, using the promotion calendar from Code 10.5 and the roster from Code 10.4, and record whether it is a horizon effect or a calendar effect wearing one.

Investigate if: the curve is flat or falling. On this data that would suggest the origins are too close together to be informative, or that the horizon is short relative to how fast this series actually changes; widen the origin spacing or lengthen the horizon and report what you changed.

Code 10.11. Demonstrate the shuffled-split failure

```
rng = np.random.default_rng(42)
hidden = rng.choice(series.index[7:-7], size=100, replace=False)

interp = (series.shift(7)[hidden].to_numpy()
          + series.shift(-7)[hidden].to_numpy()) / 2   # uses the FUTURE
actual = series[hidden]
fake = np.abs(actual.to_numpy() - interp) / actual.to_numpy()

print(f"random-day 'holdout' MAPE : {fake.mean() * 100:>6.2f}%"
      "   <- means nothing")
roll = board.loc[CHOSEN, "MAPE %"]
print(f"rolling-origin MAPE       : {roll:>6.2f}%   <- graded fairly")
print(f"sealed holiday holdout    : "
      f"{mape(holdout, fc_stat):>6.2f}%   <- the decision's own season")
```

```
print("\nthe mechanism in one line: shift(-7) hands the evaluation next")
print("week's revenue, which no deployed forecast possesses.")
```

Expected output: three MAPE figures on one screen — the shuffled pseudo-holdout, the honest rolling-origin score, and the sealed holiday score — followed by the two-line statement of the mechanism.

Input: the whole series. Transformation: one hundred randomly hidden days, each "predicted" from the seasonal lag on both sides. Output: the number an unframed pipeline would have reported, printed beside the two numbers that mean something.

The comparison is the lesson, and the three-number layout is the improvement over the draft, which showed the shuffled score alone. Placing all three side by side makes the whole chapter's evaluation argument visible in six lines of output: the shuffled score flatters, the rolling-origin score is honest about the average window, and the sealed holiday score is honest about the window the decision lives in — and they are, in that order, increasingly large and increasingly relevant. State the mechanism in writing before reading the numbers: the backward-looking term hands the evaluation next week's revenue, information no deployed forecast possesses, and each hidden day is being interpolated between a known past and a known future.

VERIFICATION CHECK

Before you run: predict where the shuffled figure will fall relative to the other two, and commit to the prediction in writing, because the point of the cell is lost if you look first.

After you run: write the audit sentence for the vendor meeting and the AI round trip alike — any forecast accuracy claim earned under shuffling is this number wearing a straight face — and then name the one-line test that unmasks it. Then write the sharper second sentence this three-way layout earns: even an honest evaluation flatters a seasonal plan if its window is chosen for convenience, which is why the holdout in this chapter was chosen for resemblance.

Investigate if: the shuffled figure is not the smallest of the three. That would mean the interpolation is not working as designed, which usually means the seasonal lag was applied in one direction only — check the sign on both shift calls.

10.12.5 Lab 10.2, Part C: The Opening, the Ramp, and the Documented Override

Part C is the chapter's verification theme at full scale: does the model see the expansion inflection? The test forecasts the known past across a store opening, so the answer can be graded. The opening is selected by rule rather than by eye, because a hand-picked break is not reproducible and may not have the four post-opening weeks the test needs.

Code 10.12. Forecast across a programmatically selected opening

```
# Choose the opening programmatically. It must sit far enough inside the
# series to have a real training window and four full post-opening weeks.
valid = opens[(opens["opening_date"]
               >= series.index.min() + pd.Timedelta(days=MIN_TRAIN))
              & (opens["opening_date"]
                 <= series.index.max() - pd.Timedelta(days=27))]
assert not valid.empty, "no opening has both a training and a test window"
break_date = valid["opening_date"].max()
store = valid.loc[valid["opening_date"].idxmax()]

others = opens.loc[opens["opening_date"] != break_date, "opening_date"]
near = others[(others - break_date).abs() <= pd.Timedelta(days=28)]
test_win = pd.date_range(break_date, periods=28, freq="D")
print(f"selected opening: {store.store_name} ({store.store_type}) "
      f"on {break_date.date()}")
print(f"other openings within 28 days: {len(near)}"
```

```
      f"   promotion days in the test window: "
      f"{int(test_win.isin(promo_days).sum())} of 28")

cut = break_date - pd.Timedelta(days=1)
tr_b, te_b = series.loc[:cut], series.loc[break_date:][:28]
for name in ["seasonal-naive", CHOSEN]:
    fc = pd.Series(fit_forecast(name, tr_b, 28), index=te_b.index)
    bias = (te_b - fc).mean()
    print(f"{name:<24s} MAPE {mape(te_b, fc):>6.2f}%"
          f"   signed error {bias:>+9,.0f}"
          f"   ({'undershoot' if bias > 0 else 'overshoot'})")
```

Expected output: the selected opening with its store type and date, the count of other openings within twenty-eight days, the number of promotion days inside the test window, and then two graded rows — seasonal-naive and the frozen method — each with MAPE, mean signed error, and a direction label; one assertion passes silently.

Input: the series and the filtered roster. Transformation: an eligibility filter on the roster, then a training cutoff at the day before the opening and a twenty-eight-day forecast across it. Output: the exhibit that teaches a planning meeting what a model can and cannot know.

The filter is the correction the draft needed. Selecting the last row of a sorted roster and adjusting by hand is not reproducible, and it can select an opening with too few post-opening days to evaluate. The rule states both requirements in code — enough history before, four full weeks after — asserts that at least one opening satisfies them, and then takes the most recent survivor. The two disclosure lines that follow are the honest half of the exercise: an opening whose window overlaps another opening or sits inside a heavy promotion is a contaminated test, and the reader is told the contamination before being shown the result rather than after.

VERIFICATION CHECK

Before you run: write the prediction in full sentences. Neither method can see the opening: seasonal-naive repeats a pre-opening week, and the frozen method extends a pre-opening level and trend. Both should therefore undershoot the post-opening series, and the signed-error column should print positive for both. Predict also whether the fitted method's undershoot will be larger or smaller than the baseline's, and why the answer is not obvious.

After you run: confirm the signs, and note what the exercise establishes — the model did not fail; the well-fitted model and the free rule failed identically in direction, because the missing information was never in the series. Then read the disclosure lines and write one sentence on how much the promotion days inside the test window complicate the reading.

Investigate if: either signed error is negative, or if the fitted method's undershoot is dramatically smaller than the baseline's. Both would suggest the model's trend term has already absorbed a run of earlier openings and is extrapolating them forward, which is a real phenomenon on this data, is the reason Code 10.14 prints a net-of-trend diagnostic, and is a finding to report rather than a problem to hide.

Code 10.13. Declare the ramp estimator and its contamination controls

```
# Two estimates of the same opening, because they answer two questions.
# RAW is the before/after step: what the series did. NET is the step the
# fitted method did NOT already extrapolate, which is the only part an
# override may add without double-counting the model's own trend.
PRE = POST = 28
MAX_PROMO = 7                    # of 28 days, counted on BOTH windows
NET_MIN_TRAIN = 180              # shortest fit the net estimate allows
```

```
def ramp_profile(d):
    """Week-by-week raw and net lift for one opening, or None."""
    before = series.loc[d - pd.Timedelta(days=PRE):
                        d - pd.Timedelta(days=1)]
    after = series.loc[d:][:POST]
    if len(before) < PRE or len(after) < POST:
        return None
    wk, a = np.arange(POST) // 7, after.to_numpy()
    out = {f"raw{w + 1}": float(a[wk == w].mean() - before.mean())
           for w in range(4)}
    train = series.loc[:d - pd.Timedelta(days=1)]
    if len(train) >= NET_MIN_TRAIN:
        resid = a - fit_forecast(CHOSEN, train, POST)
        out.update({f"net{w + 1}": float(resid[wk == w].mean())
                    for w in range(4)})
    else:
        out.update({f"net{w + 1}": np.nan for w in range(4)})
    return out


def promo_share(d):
    """Promotional days across the pre AND post windows, of 56."""
    win = pd.date_range(d - pd.Timedelta(days=PRE), periods=PRE + POST,
                        freq="D")
    return int(win.isin(promo_days).sum())
```

Expected output: nothing — this cell declares the controls and defines the ramp estimator.

Input: none. Transformation: four declared thresholds and one function. Output: the estimator Code 10.14 applies, with its assumptions written above it rather than discovered inside it. Read the function for two properties. The windows are equal in length, twenty-eight days on each side, which balances the weekday mix so that a step estimate is not partly an artifact of counting five Saturdays before and four after. And the function returns a week-by-week profile rather than a single number, because the prose of every planning deck describes new stores as ramping and an override that adds a flat step from day one contradicts the process it claims to model.

The estimator returns two profiles rather than one, and which of them the override may use is the cell's real argument. The raw profile is the before-and-after step: what the series did. The net profile is the post-opening actual minus what the fitted method itself forecast for those same days: the part the model demonstrably had not already extrapolated. A trend-tracking method fitted on a series full of recent openings has usually absorbed some of the expansion into its own trend term, so adding the raw step on top of the forecast adds that growth twice. The net profile is therefore the base adjustment and the raw profile is retained as an upper-bound sensitivity — a distinction the code enforces and the override log records, rather than a caveat the prose supplies after the fact.

Two smaller controls sit in the same cell and both close real holes. Promotional days are counted across the pre and post windows together, because a promotion inside the *before* window inflates the baseline and shrinks the apparent step exactly as much as one inside the *after* window exaggerates it. And the net estimate requires a minimum training length, because a fit on ten weeks of history is not evidence about what a method would have projected; where that minimum is not met the net columns are blank rather than invented, and the printed table shows which openings carry which estimate.

Code 10.14. Apply the exclusion rules and report every surviving estimate

```
RAW = [f"raw{i}" for i in range(1, 5)]
NET = [f"net{i}" for i in range(1, 5)]
clean, dropped = [], []
for r in opens.loc[opens["opening_date"] < break_date].itertuples():
    d = r.opening_date
    other = opens.loc[opens["opening_date"] != d, "opening_date"]
    win = pd.date_range(d - pd.Timedelta(days=PRE),
```

```
                        periods=PRE + POST, freq="D")
    p, n_promo = ramp_profile(d), promo_share(d)
    if p is None:
        why = "window falls outside the series"
    elif ((other - d).abs() <= pd.Timedelta(days=PRE)).any():
        why = "windows overlap another opening"
    elif win.month.isin([11, 12]).any():
        why = "evidence window touches the holiday season"
    elif n_promo > MAX_PROMO * 2:
        why = f"{n_promo} of 56 window days promotional"
    else:
        why = None
    if why:
        dropped.append(f"{d.date()} ({why})")
    else:
        clean.append({"opening": d.date(), "type": r.store_type,
                      "promo/56": n_promo, **p})

ramps = pd.DataFrame(clean)
assert not ramps.empty, "no uncontaminated opening evidence survives"
print("individual estimates, reported rather than averaged away:\n")
print(ramps.round(0).to_string(index=False))
print("\nexcluded:", "; ".join(dropped), sep="\n  ")

same = ramps[ramps["type"] == store.store_type]
use = same if len(same) >= 3 else ramps     # stratify where n allows
net_ok = use.dropna(subset=NET)
if len(net_ok) < 3:                         # net evidence is thinner
    net_ok = ramps.dropna(subset=NET)
profile = net_ok[NET].mean()                  # the BASE adjustment
raw_mean = use[RAW].mean()                  # upper-bound sensitivity
lo_p, hi_p = net_ok[NET].min(), net_ok[NET].max()
STRAT = "same store type" if len(same) >= 3 else "all clean types"
print(f"\nraw profile from {len(use)} openings ({STRAT});"
      f" net profile from {len(net_ok)}")
for lbl, v in [("net (base) ", profile), ("raw (upper)", raw_mean)]:
    print(lbl + " " + "  ".join(
        f"wk{i + 1} {v.iloc[i]:>+7,.0f}" for i in range(4)))
print("net range   " + "  ".join(
    f"wk{i + 1} [{lo_p.iloc[i]:+,.0f}, {hi_p.iloc[i]:+,.0f}]"
    for i in range(4)))
print("""
The gap between the rows is the trend the method already carried;
adding raw on top double-counts it. Where the net range spans zero the
evidence has not established a lift, and a scenario is the honest
instrument.""")
```

Expected output: a table of every surviving opening with its store type, promotion-day count, four weekly lift figures and its net-of-trend week-four figure; then the list of excluded openings each with its reason; then the evidence set actually used, the mean ramp profile, the range across openings at each week, and the net-of-trend mean.

Input: the roster, the promotion calendar, and the series. Transformation: four exclusion rules applied in a stated order, then stratification by store type where the count permits. Output: the override's evidence, in the form Section 10.9 requires — individual estimates visible, exclusions justified, and a range rather than a point.

Every line of the exclusion list is an argument. Openings whose windows fall outside the series cannot be measured. Openings whose windows overlap another opening cannot be separated from it. Any opening whose evidence *window* touches November or December is excluded, not only those whose own date falls there — a late-October opening measures December in its fourth week just as surely as a November opening does, and the earlier draft of this rule caught only the second kind. Windows with more than a quarter of their fifty-six days promotional

measure the promotion. An assertion then refuses to proceed on an empty table, because averaging nothing is worse than stopping.

The stratification rule does what common sense demands: a flagship and a resort store have no reason to lift equally, so when at least three openings share the type of the opening being forecast, only those are used; otherwise the wider set is used and the widening is printed. The net profile has its own fallback for the same reason, since the minimum-training rule can leave fewer net estimates than raw ones. What survives is a small number of estimates with a wide range, which is the truthful characterization of what two years of history can say about a store opening — and it is why the override in Code 10.16 carries a range, a net-versus-raw distinction, and a review date rather than a decimal.

VERIFICATION CHECK

Before you run: predict how many openings will survive the four filters, and which specific roster rows will fall to which rule. Predict the shape of the mean profile — rising across the four weeks, or flat — and say which shape would falsify the ramp story the chapter has been telling.

After you run: read the individual estimates before either mean, which is the whole point of printing them. Compare the raw row and the net row week by week and write the two sentences the gap forces: some part of the apparent step was already in the model's trend, and the size of that part is not stable across openings — so the base override is the net profile, carrying a stated range, and the raw profile is a sensitivity rather than a rival. Then ask whether a plan built on the net mean would have been safe under the lowest surviving net estimate.

Investigate if: fewer than three openings survive, if the net set falls back to mixed store types, or if the net range at week four spans zero. The first two mean the override's evidence field must say which set it rests on. The third means the evidence does not establish that openings lift revenue beyond what the method already projects, and the honest instrument is a named scenario rather than a point adjustment — which is a legitimate outcome of this lab, not a failure of it.

Code 10.15. Apply the ramp override and write the log

```
wk_of = np.clip((te_b.index - break_date).days // 7, 0, 3)
fc_b = pd.Series(fit_forecast(CHOSEN, tr_b, 28), index=te_b.index)
day = (te_b.index - break_date).days
w1, w4 = day < 7, day >= 21
flat = float(np.nanmean(profile.to_numpy()))
for name, fc in [
        (CHOSEN, fc_b),
        ("+ flat step (net mean)", fc_b + flat),
        ("+ net ramp (base)", fc_b + profile.to_numpy()[wk_of]),
        ("+ raw ramp (upper)", fc_b + raw_mean.to_numpy()[wk_of])]:
    print(f"{name:<24s} MAPE {mape(te_b, fc):>6.2f}%"
          f"   wk1 {(te_b - fc)[w1].mean():>+8,.0f}"
          f"   wk4 {(te_b - fc)[w4].mean():>+8,.0f}")
print("""
The flat step and the net ramp share a mean by construction and differ
in shape. Read the raw row as a sensitivity, not as a rival: whatever
advantage it shows here is bought by re-adding growth the method had
already extrapolated, and it would not survive on a series where the
method had not.""")

print(f"""
OVERRIDE LOG -- {break_date.date()} {store.store_name} opening
 1 change     the 28 forecast days from {te_b.index.min().date()},
              ADJUSTED by a week-specific ramp, not a flat step
```

```
 2 size       {profile.iloc[0]:+,.0f}/day in week 1 to
              {profile.iloc[3]:+,.0f}/day in week 4, net of the trend
              the method already carried; an early week may be negative
 3 evidence   {len(use)} controlled openings ({STRAT}); the net
              profile rests on {len(net_ok)}; week-4 net range
              [{lo_p.iloc[3]:+,.0f}, {hi_p.iloc[3]:+,.0f}]; raw week-4
              step {raw_mean.iloc[3]:+,.0f}, carried as the upper-bound
              sensitivity and as the upside scenario, not as the base
 4 owner      <your name>, analyst of record
 5 review     {(break_date + pd.Timedelta(days=28)).date()}, when the
              four weeks of actuals mature
 6 baseline   the unadjusted {CHOSEN} row above, preserved""")
```

Expected output: three graded rows — the frozen method, the same method plus a flat-step override, and the same method plus the ramp override — each with MAPE and the signed error in week one and week four; a two-line note on why the means coincide; and the six-element override log.

Input: the pre-opening forecast, the ramp profile, and the test window. Transformation: a week-indexed adjustment. Output: the analyst's contribution, measured — and then written down in the form that makes it auditable.

The comparison is designed to isolate shape from size, which is why it reports week one and week four separately rather than one pooled bias. A flat step and a ramp built from the same evidence share a mean by construction, so a pooled signed error cannot distinguish them; the first and last weeks can. Four rows appear rather than three, and the fourth is the honest one: the raw ramp is shown beside the net ramp so the reader can see what the upper bound would have done. Read that row as a sensitivity rather than a rival. Whatever advantage it shows is bought by re-adding growth the method had already extrapolated, and it would evaporate on a series where the method had not — which is exactly why the base override is the net profile and the raw profile becomes the upside scenario of Code 10.20 rather than the recommendation.

The log printed at the end is the deliverable, not the code. Read its six fields against Section 10.9 and note what each one costs to fill: the size field required a profile, the evidence field required the exclusions and the range and the net-of-trend figure, the review date required a decision about when this adjustment will be graded, and the baseline field required keeping the unadjusted row on the leaderboard where January can find it. An override log is expensive to fill honestly and worthless filled dishonestly, which is why the chapter prints one rather than describing one.

VERIFICATION CHECK

Before you run: predict whether the net ramp will beat the flat step on MAPE, and predict the direction of each override's week-one and week-four errors. Then predict where the raw ramp will land relative to both, and write down in advance why a better MAPE from the raw row would not make it the right choice.

After you run: confirm that the flat step and the net ramp share a pooled mean and that their weekly errors differ, then write the sentence that distinguishes them. State the net override's measured contribution in MAPE points and in dollars across the four weeks, because that figure is the analyst's own graded output. Then fill the log's owner field with your name and the review date in your calendar, and mean both.

Investigate if: the net ramp does not beat the flat step, or the net profile is negative in an early week. The designed expectation is that shape helps and that the raw row flatters itself; a different result is a finding to trace, not an error to edit away. Check whether the promotion days disclosed in Code 10.12 sit in the first or the last week, since a promotion in week one will make a flat step look better than it is — and remember that a negative early-week net adjustment is legitimate, which is why the log says the forecast was adjusted rather than raised.

10.12.6 Lab 10.2, Part D: Intervals That Aggregate in the Right Order

Part D produces the uncertainty the plan consumes. Three cells build it, and the middle one contains the single most important arithmetic correction in this chapter.

Code 10.16. Build and grade the daily prediction interval

```
REPS = 2000
selected_fit = fit_candidate(CHOSEN, select)


def empirical_paths(name, train, h, reps, seed=42):
    """Declared fallback when a zero-parameter rule wins: resample whole
    rolling-origin error paths, so the day-to-day dependence travels with
    the draw. Recycles beyond H and says so."""
    errs = np.array([
        series.loc[o + pd.Timedelta(days=1):][:H].to_numpy()
        - fit_forecast(name, series.loc[:o], H) for o in origins])
    g = np.random.default_rng(seed)
    draw = errs[g.integers(0, len(errs), reps)][:, np.arange(h) % H].T
    return pd.DataFrame(fit_forecast(name, train, h)[:, None] + draw,
                        index=holdout.index)


if selected_fit is None:
    INTERVAL_SOURCE = f"empirical rolling-origin residuals ({CHOSEN})"
    sims = empirical_paths(CHOSEN, select, len(holdout), REPS)
else:
    # statsmodels 0.14.6 spells the seed random_state=; 0.15 renames it.
    INTERVAL_SOURCE = f"{CHOSEN} simulation"
    sims = selected_fit.simulate(len(holdout), anchor="end",
                                 repetitions=REPS, error="add",
                                 random_state=42)
assert sims.shape == (len(holdout), REPS)
print("interval source:", INTERVAL_SOURCE, " paths:", sims.shape)

lo, hi = sims.quantile(0.10, axis=1), sims.quantile(0.90, axis=1)
inside = (holdout >= lo) & (holdout <= hi)
width = hi - lo
print(f"\nnominal coverage    80.0%")
print(f"realized coverage   {inside.mean():>5.1%}"
      f"   ({int(inside.sum())} of {len(holdout)} days)")
print(f"mean width          {width.mean():>9,.0f}"
      f"   ({width.mean() / holdout.mean():.0%} of mean daily revenue)")
print(f"evaluated periods   {len(holdout)} days from ONE forecast origin")

blk = np.arange(len(holdout)) // 7
full = blk < len(holdout) // 7               # the last days are a stub
diag = pd.DataFrame({"block": blk + 1, "inside": inside.to_numpy(),
                     "width": width.to_numpy()})[full]
print("\nby horizon week (7-day blocks, stub excluded):")
print(diag.groupby("block")
      .agg(days=("inside", "size"), coverage=("inside", "mean"),
           width=("width", "mean")).round(2).to_string())
print(f"stub: {int((~full).sum())} days beyond the eighth full week")
```

Expected output: the simulated array's shape and date span, then the nominal and realized coverage with the count of days behind it, the mean interval width in dollars and as a share of mean daily revenue, the number of evaluated

periods and their single origin, and then coverage and width by seven-day block with the stub disclosed; one assertion passes silently.

Input: the fitted model and the sealed holdout. Transformation: two thousand simulated future paths, then daily percentiles across them. Output: the daily 80% band, and the five figures that constitute an honest grade of it.

The first line of the cell is the one that keeps the whole chapter honest. The band is built from the method the comparison actually selected, through the same factory Code 10.7 declared — not from a separately typed Holt-Winters call that would silently replace the winner. And because the two baselines fit nothing, the cell has to say in advance what it will do if a zero-parameter rule wins: it falls back to resampling that rule's own rolling-origin error paths, whole, so the day-to-day dependence travels with the draw, and it prints which source produced the band. A notebook that quietly substituted a model with a convenient simulate method would be reporting an interval that belongs to a method nobody chose.

The reporting standard is the second correction. A single coverage percentage cannot be read as a verdict, so the cell prints the whole set Section 10.8 specified — nominal, realized, mean width, period count, and coverage by horizon — and it labels the period count with the fact that all of them come from one forecast origin, which is the qualification that keeps a reader from treating sixty-one dependent daily observations as sixty-one independent tests. Read the by-week block with that in mind: seven days per block is far too few to estimate a coverage rate, so the column is a pattern to investigate rather than a set of measurements, and the width column beside it is the more informative of the two because it shows the cone opening.

VERIFICATION CHECK

Before you run: predict the realized coverage and, separately, predict what the width column will do across the eight blocks. Then predict something harder: given what Code 10.9 showed about this method on this season, predict whether the misses that fall outside the band will be balanced or one-sided.

After you run: read all five figures together before drawing any conclusion, then write the paragraph the deliverable needs. If realized coverage is near nominal, say what that does and does not license: it says the band prices day-to-day noise about right; it says nothing about the calendar, because a band centered on a forecast that systematically undershoots a season can still contain most individual days while being wrong about the total. If coverage is materially off in either direction, say which and record the width beside it, because a coverage figure quoted without a width can be bought by widening the band.

Investigate if: coverage is far above nominal and the mean width is a large share of mean daily revenue. That combination is the over-wide band the plan will be over-hedged by, and the response is to check the fitted error assumption against the remainder panel from Code 10.4 before shipping it.

Code 10.17. Aggregate paths into weekly intervals

```
# Weekly bins are declared, not inherited from a resample default: eight
# consecutive 7-day blocks from the forecast origin, plus a named stub.
labels = {b: str((holdout.index[b * 7]).date()) for b in np.unique(blk)}
tag = pd.Series([labels[b] for b in blk], index=holdout.index)

# WRONG: seven daily 10th percentiles added together is not the 10th
# percentile of the seven-day total.
wrong_lo, wrong_hi = lo.groupby(tag).sum(), hi.groupby(tag).sum()

# RIGHT: aggregate each simulated PATH first, then take quantiles across
# the weekly totals, which respects the dependence among the days.
wk_sims = sims.groupby(tag).sum()
right_lo = wk_sims.quantile(0.10, axis=1)
```

```
right_hi = wk_sims.quantile(0.90, axis=1)

cmp = pd.DataFrame({"actual": holdout.groupby(tag).sum(),
                    "forecast": wk_sims.mean(axis=1),
                    "lo80": right_lo, "hi80": right_hi,
                    "width_right": right_hi - right_lo,
                    "width_wrong": wrong_hi - wrong_lo})
print(cmp.round(0).to_string())
ratio = cmp["width_wrong"].mean() / cmp["width_right"].mean()
shuffle_rng = np.random.default_rng(42)      # local, not inherited
indep = sims.to_numpy().copy()               # destroy the dependence
for i in range(indep.shape[0]):
    shuffle_rng.shuffle(indep[i])
ind = pd.DataFrame(indep, index=sims.index).groupby(tag).sum()
ind_w = float((ind.quantile(0.90, axis=1)
               - ind.quantile(0.10, axis=1)).mean())
print(f"\nsummed edges are {ratio:.2f}x the correct width HERE. Shuffle "
      "the day-to-day\ndependence away and the same shortcut gives "
      f"{cmp['width_wrong'].mean() / ind_w:.2f}x. A quantile of a sum is "
      "not\nthe sum of the quantiles; the direction and the size of the "
      "error depend\non the marginals and their dependence, so neither "
      "can be reasoned about\nwithout the paths.")
```

Expected output: a nine-row table of declared seven-day blocks with the actual total, the forecast total, the correct 80% bounds, and the widths produced by the correct and the incorrect method side by side; then the ratio between them, and the same ratio recomputed after the day-to-day dependence has been shuffled away.

Input: the simulated paths and the holdout. Transformation: two aggregations of the same simulations in two different orders. Output: the correction of Section 10.8, made numerical.

The wrong method sums the daily bounds; the right method sums each path and then takes percentiles across the path totals. Note what the cell does not do: it does not merely assert that the second is correct. It computes both, prints the ratio, and then computes the ratio a third time on shuffled paths where the dependence has been destroyed. On the real simulations the summed-edges band is only a few percent too wide, because Holt-Winters paths drift together — a path that runs high runs high all week. On the shuffled version the same formula overstates the width by a factor near the square root of seven. Read the general claim carefully, because it is weaker and more useful than "summing makes the band too wide": a quantile of a sum is not the sum of the quantiles, and the direction as well as the size of the discrepancy depends on the marginals and their dependence. Widening happens to be the direction here; another dependence structure need not produce it. That is why "the error is small in practice" is not a defense — the analyst who used the shortcut had no way to know how small, or which way.

The weekly bins are also declared rather than inherited. A resample to calendar weeks would cut the first and last blocks short and let a library default decide which days the plan's weeks contain; the cell instead counts seven-day blocks from the forecast origin, labels each by its start date, and discloses the stub of days left over at the end. When the managerial calendar requires weeks ending on a particular weekday, that is a separate declaration to be aligned with the reporting calendar on purpose — not a default to be discovered afterward.

VERIFICATION CHECK

Before you run: predict the direction of the error in the summed-edges band — too wide or too narrow — and, more importantly, write down whether you believe that direction is general or particular to this model. Then predict the size of the ratio on the real simulations, and predict how it will change when the dependence is shuffled away.

After you run: write the two sentences this cell exists to produce. First, the arithmetic one: a quantile of a sum is not the sum of the marginal quantiles, and the sum of seven daily tenth percentiles corresponds to all seven days landing at their own tenth percentiles at once, which is a much rarer event than the week's total landing at its tenth. Second, the epistemic one: the direction and the size of that discrepancy depend on the marginals and their dependence, so neither can be estimated without the paths — which means the shortcut is not a small approximation but an unquantified one.

Investigate if: the two widths are nearly identical. That would indicate near-total dependence across days within a path, which is worth confirming against the simulated paths themselves — and if it holds, it is a fact about the fitted model worth stating in the deliverable, because it means the weekly band and the daily band carry nearly the same information.

Code 10.18. Read the season total and plot the cone

```
season = sims.sum(axis=0)                 # one season total per path
s_lo, s_hi = season.quantile(0.10), season.quantile(0.90)
print(f"\nseason total across {REPS} paths:"
      f"   lo80 {s_lo:>11,.0f}   mean {season.mean():>11,.0f}"
      f"   hi80 {s_hi:>11,.0f}")
print(f"actual season total: {holdout.sum():,.0f}"
      f"   inside the band: "
      f"{bool(s_lo <= holdout.sum() <= s_hi)}")

fig, ax = plt.subplots(figsize=(9, 4))
ax.plot(select.index[-90:], select.iloc[-90:].to_numpy(), lw=0.8,
        color="0.4", label="history")
ax.plot(holdout.index, holdout.to_numpy(), lw=0.9, color="black",
        label="actual")
ax.plot(sims.index, sims.mean(axis=1).to_numpy(), lw=1.3,
        color="#57068C", label="forecast")
ax.fill_between(sims.index, lo.to_numpy(), hi.to_numpy(), alpha=0.20,
                color="#57068C", label="80% daily band")
ax.legend(fontsize=8, loc="upper left")
ax.set_title("the cone and what it omits", fontsize=9, loc="left")
fig.tight_layout()
```

Expected output: the season total's 80% bounds and mean across the simulated paths, the actual season total, whether it fell inside the band, and a forecast-cone figure showing history, actual, forecast, and the daily band.

Input: the simulated paths. Transformation: one sum per path, then percentiles across the totals. Output: the season-total range the plan consumes, and the cone that shows a meeting what a fanning interval looks like.

This cell is where the qualification of Section 10.8 stops being rhetorical. The daily band, as Code 10.16 showed, achieved close to its nominal coverage — and yet the season total lands high in its range, near the upper bound, because the forecast the band is centered on undershoots a holiday it was never told about. Both facts are true and they are not in tension: the band prices day-to-day innovation faithfully and prices the missing calendar not at all. Write that sentence into the deliverable verbatim, because it is the single most useful thing this chapter can tell a planning meeting about what an interval is. An interval is a statement about the future under the model. When the model is missing a component the business knows about, the interval is silent about exactly the risk the business is most exposed to — and the scenario table of Code 10.20 exists to carry what the band cannot.

10.12.7 Lab 10.2, Part E: The Planning Forecast, the Envelope, and the AI Round Trip

Everything so far has forecast the known past, so that it could be graded. Part E forecasts the future the commission actually asked about, which cannot be graded at all — and the discipline of the section is saying so clearly while still producing a number the plan can use.

Code 10.19. Refit on all history and declare the analyst's calendar

```
# Refit on ALL certified history, then forecast the horizon the decision
# spans. No actuals exist here; this is the deliverable, not a grade.
full_fit = fit_candidate(CHOSEN, series)      # the SELECTED method
assert full_fit is not None or CHOSEN in MODEL_CONFIG, (
    "a zero-parameter rule won: build the plan from empirical_paths "
    "and say so in the deliverable")
plan_idx = pd.date_range(PLAN_START, PLAN_END, freq="D")
stat_fc = pd.Series(fit_forecast(CHOSEN, series, len(plan_idx)),
                    index=plan_idx)

# THE ANALYST'S CALENDAR. None of this is in the series.
ANNOUNCED = pd.DataFrame({
    "store_name": ["Philadelphia", "Short Hills"],
    "store_type": ["Flagship", "Suburban"],
    "opening_date": pd.to_datetime(["2026-10-05", "2026-10-19"])})
HOL = (pd.Timestamp("2026-11-01"), pd.Timestamp("2026-12-31"))

IDX25 = holiday_index(series, 2025, CHOSEN)
both = pd.concat([IDX24.rename("2024"), IDX25.rename("2025")], axis=1)
print("holiday index by day from Thanksgiving -- two observations only:")
print(f"  peak 2024 {IDX24.max():.2f}   peak 2025 {IDX25.max():.2f}"
      f"   correlation {both.corr().iloc[0, 1]:.2f}")
```

Expected output: nothing but the holiday-index comparison — the peak index in each of the two available years and the correlation between them.

Input: the full certified series. Transformation: a refit on all history, a forecast across the planning horizon, and a second holiday index estimated from the 2025 season. Output: the statistical spine of the planning forecast, and the announced calendar the analyst brings to it.

The refit is on all certified history, because the sealed holdout has done its job and withholding data from the forecast that will actually be used would be superstition rather than discipline. The announced openings are declared as a small table in the notebook, which is exactly right: they are not in the series, they cannot be in the series, and the only place they can enter is a line of code an analyst wrote and signed. And the two holiday indices are printed side by side because that comparison is the whole evidence base for the annual layer. Two observations, one of them contaminated by an opening, correlated but not identical: that is what StyleCraft knows about its own holiday shape, and it is why the next cell produces three scenarios rather than one forecast.

Code 10.20. Build the holiday-season scenario envelope

```
# Coherent whole-year profiles, not a day-by-day minimum: the downside
# is the weaker observed holiday, not a path stitched from whichever
# year was lower at each offset. The ramp assumption moves with it.
off26 = pd.Series((plan_idx - thanksgiving(2026)).days, index=plan_idx)
in_hol = plan_idx.to_series().between(*HOL)
strength = both.mean(axis=0)
down_year, up_year = strength.idxmin(), strength.idxmax()
net_v, raw_v = profile.to_numpy(), raw_mean.to_numpy()
SCEN = {"downside": (both[down_year], net_v * 0.6, "net x0.6"),
        "base":     (both.mean(axis=1), net_v, "net"),
        "upside":   (both[up_year], raw_v, "raw upper")}
print(f"holiday profiles: downside {down_year}, upside {up_year}"
      f"  (mean index {strength[down_year]:.2f} vs"
      f" {strength[up_year]:.2f})")


def build(idx, ramp_vec):
    fc = stat_fc * off26.map(idx).ffill().bfill().where(in_hol, 1.0)
    for r in ANNOUNCED.itertuples():          # the ramp override, forward
```

```
        wk = (fc.index - r.opening_date).days // 7
        fc = fc + np.where(wk < 0, 0.0, ramp_vec[np.clip(wk, 0, 3)])
    return fc


plan = {k: build(i, v) for k, (i, v, _) in SCEN.items()}
season = {k: v.loc[HOL[0]:HOL[1]] for k, v in plan.items()}

# Preserve the fitted model's ADDITIVE error structure: centre the
# simulated paths and ADD them, rather than turning them into
# proportional shocks that grow with each scenario's level.
sim = full_fit.simulate(len(plan_idx), anchor="end", repetitions=400,
                        error="add", random_state=7).to_numpy()
innov = sim - sim.mean(axis=1, keepdims=True)
rows = []
for k, fc in plan.items():
    tot = (fc.to_numpy()[:, None] + innov)[in_hol.to_numpy()].sum(axis=0)
    rows.append({"scenario": k, "point": season[k].sum(),
                 "lo80": np.quantile(tot, .10),
                 "hi80": np.quantile(tot, .90),
                 "peak index": SCEN[k][0].max(), "ramp": SCEN[k][2]})
scen = pd.DataFrame(rows).set_index("scenario").round(2)
print("\nholiday-season revenue, Nov 1 - Dec 31 2026, assumptions shown:")
print(scen.to_string())

pt, base = scen["point"], scen.loc["base"]
half = (pt.max() - pt.min()) / (2 * pt.loc["base"])
sim = (base["hi80"] - base["lo80"]) / 2 / pt.loc["base"]
print(f"\nscenario envelope {pt.min():,.0f} to {pt.max():,.0f}"
      f"  (+/-{half:.0%} of base)")
print(f"simulation band   {base['lo80']:,.0f} to {base['hi80']:,.0f}"
      f"  (+/-{sim:.0%} of base)")
print("""
CAUTION. Rolling origins graded this method to 56 days; December sits
three times further out. Report both widths, say which dominates, and
label the envelope an assumption set rather than a graded forecast.""")
```

Expected output: a three-row scenario table with the Q4 point total, the 80% simulation bounds, and the two named assumptions behind each row; then the scenario envelope and the simulation band as separate ranges with their half-widths; then a five-line caution about the horizon.

Input: the refitted forecast, the two holiday indices, the announced openings, and the ramp profile from Code 10.14. Transformation: three scenarios, each combining a holiday index drawn from the available evidence with a multiplier on the opening ramp. Output: the planning range, and an explicit statement of which instrument carries it.

The scenarios are constructed from the evidence rather than from adjectives, and two details in that construction are easy to get wrong. The first is coherence. Taking a day-by-day minimum across the two available holiday indices would not produce "the weaker holiday"; it would produce a synthetic path stitched from whichever year happened to be lower at each offset — a season that never occurred and that no argument supports. The cell instead ranks the two observed profiles by their mean strength and uses whole years: the downside is the weaker holiday as it actually unfolded, the upside the stronger one, the base their average. The second is what moves with the holiday assumption. The ramp assumption travels with it, from a damped net override in the downside, through the net profile in the base, to the raw upper bound in the upside — so each row is one internally consistent world rather than a mix of assumptions from different ones. Both assumption columns are printed, which is the difference between a scenario set and three guesses wearing a rubric.

The uncertainty around each row also has to respect the model it came from. Holt-Winters was fitted with additive errors, so its simulated disturbances are dollar amounts, not percentages. Rescaling the simulated paths into multiplicative shocks — dividing by their own mean and multiplying by each scenario's level — would quietly convert them into proportional errors that grow with the scenario, which is a different model from the one that was

graded. The cell centres the simulated paths and adds them, which preserves the fitted structure and keeps the three bands comparable.

The caution at the end is the cell's most important output and the most easily omitted. The rolling-origin evaluation graded this method to fifty-six days. December 2026 sits roughly one hundred eighty days from the forecast origin, three times the graded horizon, and the simulation band there has widened to a range no plan can act on. So the deliverable says which instrument is the planning range at which horizon: the band for the near weeks, the scenario envelope for the season — and the envelope is labeled as an assumption set rather than a graded forecast, with a monthly re-forecast cadence and a named trigger, the October actuals, at which the whole table is reopened.

VERIFICATION CHECK

Before you run: predict the ordering of the three scenarios (trivial) and then the harder thing — predict the ratio between the scenario envelope's half-width and the simulation band's half-width, and say which one you expect to be larger and why. Note that the half-width printed here is the envelope's span divided by twice the base case, which is a symmetric width around the base rather than a ratio of the extremes; a ratio of maximum to minimum is a different number and answers a different question.

After you run: write the paragraph the plan will actually consume. Name the base case, the envelope, the two assumptions that move it, and the horizon caveat, in that order. Then answer the question a planning lead will certainly ask: if the band is this wide, why is the point forecast worth anything at all? The honest answer is in Section 10.13 and it is worth rehearsing here — the point anchors the level and the weekly shape, both of which were graded; the range around it prices what was not.

Investigate if: the downside scenario falls below the prior year's actual season total. That would mean the downside is asserting a decline the evidence does not support, and its assumption column should be revised until it does. Do not treat a scenario envelope wider than the simulation band as a defect. Compare the two widths and disclose which source of uncertainty dominates: at a six-month horizon the calendar and structural assumptions may reasonably do more of the planning work than the fitted error distribution, and saying so is the deliverable's job rather than a problem to be tuned away.

Code 10.21. Translate the forecast into the open-to-buy envelope

```
# A DOLLAR envelope, every assumption declared and varied by scenario.
# Not a unit buy -- see the scope note below.
OTB = {"downside": (0.14, 0.52, 0.55),   # markdown, cost, Jan index
       "base":     (0.10, 0.50, 0.62),
       "upside":   (0.07, 0.48, 0.70)}
BOM_RETAIL = 1_150_000.0     # inventory on hand at retail, November 1
WOS_EOM = 4.0                # weeks of supply carried into January
MEDIA_LEAD_DAYS = 7          # declared: spend leads the demand it buys

rows = []
for k, fc in season.items():
    md, cost_pct, jan = OTB[k]
    sales = float(fc.sum())
    eom = fc.mean() * 7 * jan * WOS_EOM
    net = sales + eom + sales * md - BOM_RETAIL     # may be negative
    rows.append({"scenario": k, "holiday sales": sales,
                 "markdowns": sales * md, "EOM inventory": eom,
                 "net receipts": net,
                 "receipts (retail)": max(net, 0.0),
                 "excess inventory": max(-net, 0.0),
                 "open-to-buy (cost)": max(net, 0.0) * cost_pct})
```

```
otb = pd.DataFrame(rows).set_index("scenario").round(0)
print("open-to-buy envelope, Nov-Dec 2026 -- dollars, not units:")
print(otb.to_string())
b = otb["open-to-buy (cost)"]
wide = (f"({b.max() / b.min() - 1:.0%} wide)" if b.min() > 0 else
        "-- the downside needs no new receipts, which is a finding for"
        "\nthe buying team in August rather than an error")
print(f"\nenvelope {b.min():,.0f} to {b.max():,.0f}  {wide}")

wk = season["base"].groupby(
    np.arange(len(season["base"])) // 7).agg(["sum", "size"])
pace = pd.DataFrame({"revenue": wk["sum"].to_numpy(),
                     "days": wk["size"].to_numpy()})
pace["share"] = pace["revenue"] / pace["revenue"].sum()
pace["spend block"] = [
    str((season["base"].index[i * 7]
         - pd.Timedelta(days=MEDIA_LEAD_DAYS)).date())
    for i in range(len(pace))]
pace["paced spend"] = (pace["share"] * 480_000.0).round(0)
print(f"\nmedia pacing, base case, {MEDIA_LEAD_DAYS}-day lead:")
print(pace.set_index("spend block").round(3).to_string())
print("""
A demand-shaped pacing heuristic, not a causal or ROI-optimized
allocation; reconcile it with the campaign calendar before committing.

SCOPE. Revenue and open-to-buy dollars only. Assortment- and store-
level unit buys need a separate disaggregated forecast at those
grains, with its own baselines and its own holdout.""")
```

Expected output: a three-row open-to-buy table with Q4 sales, planned markdowns, ending inventory, receipts at retail and open-to-buy at cost; the envelope span; a nine-row media pacing table with revenue, days, share of the season and paced spend per declared block; and the scope statement.

Input: the three Q4 scenarios. Transformation: a declared open-to-buy identity and a declared media-pacing share. Output: the deliverable's planning translation, and the boundary the opening case promised.

Read the open-to-buy arithmetic as a chain of declared assumptions rather than a formula. Net receipts at retail are planned sales plus planned ending inventory plus planned markdowns minus the inventory already on hand, and each of the four terms is either forecast or declared: sales come from the scenario, markdowns from a declared rate that varies by scenario, ending inventory from a declared weeks-of-supply target applied to a declared January index, and beginning inventory from the current position. The cost complement converts receipts at retail into dollars of buying authority. Every one of those numbers is visible, every one varies by scenario, and none of them is a unit count — which is the point of the scope statement at the end.

One arithmetic detail is a business fact rather than a rounding decision. The identity can return a negative number, and a negative result is meaningful: it says the inventory already on hand more than covers the scenario's planned sales, markdowns and ending position. It is not, however, negative buying authority. The cell therefore separates the diagnostic from the operational amount — reporting net receipts as computed, the operational receipts floored at zero, and the implied excess inventory as its own column — and the spread line refuses to divide by a lower edge of zero. A zero lower edge is a finding the buying team needs in August, phrased as such, rather than a percentage that fails to compute.

The media-pacing table carries two labels it must not lose. It is a demand-shaped pacing heuristic, not a causal or ROI-optimized allocation: it spends in proportion to forecast revenue and establishes nothing about what that spend causes, which is Chapter 11's question rather than this one's. And because Section 10.9 said media should lead the demand it buys, the table applies a declared lead — the shares are computed on the revenue blocks and reported against spend blocks shifted earlier by that many days — so the lead is a visible parameter someone can argue with rather than an implicit assumption that spend and demand land in the same week.

That scope statement is the answer to the question the audit of this chapter asked hardest. A chain-level revenue forecast can size a dollar envelope and pace media by week. It cannot tell a buyer how many units of which style in which size go to which store, because that translation needs realized price, category and size mix, margin, markdown expectation, and store allocation — none of which live in a daily revenue series. So the chapter declares the envelope, hands the buying team a dollar figure with a range and a named set of assumptions, and names the disaggregated forecast at category and store grain as separate work with its own baselines and its own holdout. Promising quantities the analytical frame cannot produce is a specific and common way for an honest forecast to become a laundered guess, and Section 10.15 will say so in those words.

VERIFICATION CHECK

Before you run: predict whether the open-to-buy envelope will be proportionally wider or narrower than the revenue envelope that produced it, and explain the mechanism before you look. Then predict which declared assumption the envelope is most sensitive to, and check your prediction afterwards by changing one number at a time.

After you run: run that sensitivity check for real. Move the markdown rate, the weeks-of-supply target, and the cost complement one at a time and record which one moves the envelope most, because that is the assumption the deliverable must foreground and the one the planning meeting will argue about. Then read the media pacing table against the cone from Code 10.18 and write the pacing sentence: which weeks lead demand, and how much of the budget the plan can commit before the October re-forecast.

Investigate if: any scenario's net receipts come back negative, so that the operational floor and the excess-inventory column both engage. That is arithmetically possible and operationally meaningful — it says that scenario needs no new receipts at all, which is a finding the buying team must hear in August rather than in November. Check also that the declared media lead is compatible with the campaign calendar from Code 10.5 before anyone paces against it.

The lab closes with the AI round trip. Hand an assistant the raw marketing_daily file and the deliberately loose prompt "forecast our daily revenue for the next eight weeks — maximize accuracy," and grade the response in writing with Table 8.5's five points followed by Table 10.5's four. Find the split and run the date-order check on it; the designed likelihood, per Section 10.11, is shuffling or a single reused block. Determine whether method selection and the reported grade happened on the same data, which is the failure the draft of this chapter committed and the audit caught. Reconstruct what the pipeline did about the openings, the promotions, and the holiday it was never told about — the answer will be nothing. Check every reported metric for the near-zero denominator and every forecast for its missing interval, and check any weekly or total interval for whether it was built by summing daily bounds. Then compare its headline number against the rolling-origin band you predicted before it ran, and against the sealed-holiday number, which is the one the plan would actually have lived in. Re-prompt with the full frame per the AI in Practice box and compare the two responses. Your audit memo, filed per Appendix D, is the lab's final deliverable — and the direct rehearsal of the vendor-button verdict Section 10.13 writes.

10.13 Marketing Interpretation and Managerial Insight

The lab's outputs are leaderboards, cones, and a log; the commitments run on sentences. This section translates — and, per the guide's standing practice, it does so partly by exhibiting the wrong managerial readings and correcting them, because forecasting mints a special class of misreading: its numbers sound like promises, its percentages sound like guarantees, and both will be spoken in the planning meeting by people whose only error is treating a probability statement as a commitment.

The first wrong reading is the one the leaderboard invites, and it will be voiced the moment the leaderboard goes up: "The model's MAPE is 9 percent. So we'll be within 9 percent in December — plan accordingly." Every

clause is wrong in a way a section of this chapter has priced, and this chapter's evaluation design was built so that the correction is a number rather than a caution. A MAPE is an average, not a bound: the evaluated days missed by more and less, and roughly half the misses exceeded the average — the interval, not the MAPE, states the range. It is an average at measured horizons: December sits at the far end of the cone, where Code 10.10's horizon table showed error running well above the pooled number, and the December weeks deserve the week-seven-and-eight grade. It was measured on the wrong season: the 9 percent came from rolling origins across ordinary spring and summer windows, and the same frozen method, graded on the sealed November–December holdout, missed by close to twice that — in one direction. And a percentage of December is the largest dollar figure of the year: nine percent of the season's peak weeks is a different commitment problem than nine percent of a quiet April, which is why the deliverable translates every accuracy figure into dollars at the season's scale before the meeting does it badly. The corrected sentence, worth writing on the leaderboard itself: the method's rolling-origin error is X percent, its error on the last holiday season it was graded against was Y percent with a systematic undershoot, and the planning range for this December is the scenario envelope, not either percentage.

The second wrong reading is quieter and does more organizational damage: the forecast read as a commitment. "You forecast $1.7 million and we got $1.5 — you were wrong, and planning built the receipts on your number." The correction has two halves, and the deliverable must make both in advance, not in January. The first half is Section 10.8's: the forecast was a distribution summarized by a point, the range travelled with it, and a forecast is wrong when actuals fall outside the communicated range more often than the stated rate — not whenever the point misses, because the point always misses. The second half is Section 10.9's word discipline: the moment the midpoint was copied into the plan as a commitment — and the range quietly deleted in the copying — the organization converted a probability statement into a promise nobody made. The deliverable's defense is structural: the range travels with the number in every artifact, the scenario table names what would move the outcome to each edge, and the planning translation states explicitly which side of the range each decision leaned toward and why.

The third wrong reading is the one this chapter's own draft invited, and correcting it is the reason Section 10.1 put a boundary in the brief. "The forecast says $1.7 million, so buy $1.7 million of product." A revenue forecast is not a buy. Between the two sit realized price, category and size mix, gross margin, expected markdowns, inventory already on hand, sell-through, and store allocation — every one of them an assumption, none of them present in a chain-level daily revenue series. What the deliverable can honestly produce is what Code 10.21 produces: an open-to-buy envelope in dollars, built from the forecast by a declared identity whose four terms are visible and vary by scenario, with the assumption the envelope is most sensitive to named. What it cannot produce is a unit plan, and saying so is not a hedge — it is the difference between handing the buying team a number they can interrogate and handing them a number that will be interrogated in January when it is too late. The unit-level forecast is real work at a different grain, with its own baselines and its own holdout, and naming it in the deliverable is the analysis pricing its own upgrade.

The deliverable that survives all three misreadings has a fixed anatomy, assembled entirely from lab outputs. Page one is the frame and the calendar: the series definition with its reconciliation certificate from Code 10.3, the three declared windows with their dates and their different rights per Table 10.3, and the known-calendar list of Section 10.4 — openings, promotions, and the date of Thanksgiving in every year the analysis touches — with each forecast-window item matched to an override or a stated assumption. The rolling-origin leaderboard follows, per Section 8.9's fixed rules: naive, seasonal-naive, Holt-Winters, and damped Holt-Winters on the same origins, in MAPE with minimum denominator and MAE beside it, with the week-eight column and the bias column shown and the margin over the opponent stated in dollars per day. Then the sealed holdout, once: the frozen method and the frozen method plus the calendar layer, graded on a prior holiday season, with the signed error and the season-total miss — the exhibit that tells the meeting what the model cannot know. Then the horizon table with its error counts. Then the inflection exhibit from Codes 10.12 through 10.15 — both methods undershooting the opening, and the net ramp override closing the shape as well as the level, with the raw profile shown beside it as the upper bound it is — because it is the one exhibit that inoculates the room against the vendor button's confidence. Then the override log, six elements per entry, with the net-versus-raw distinction in the evidence field and the statistical baseline preserved. Then the interval paragraph: the daily coverage with its width and its period count, the weekly and season-total bands built path-first, and the sentence about what the band does not price. Then the scenario table, one internally

consistent world per row, with both assumption columns, the scenario envelope and the simulation band reported side by side, and a sentence naming which of the two dominates at this horizon. And finally the planning translation: the open-to-buy envelope at each edge with its declared assumptions and its sensitivity, the asymmetry of Section 2.7 argued explicitly — markdown cost against stockout cost, and which side the recommendation therefore leans — the media pacing shape at the declared weekly blocks, the scope statement naming what the envelope is not, and the re-forecast calendar: who reruns this, on what cadence, and which meeting reopens it when October actuals land. A forecast, in this guide's definition, is all of that; the number alone is just the part that fits in a cell.

10.14 Business Analytics in Practice

This section turns from the fictional case to how forecasting operates in demand planning, disruption management, and media pacing — where its outputs move physical goods and real budgets, and where the hardest-won lessons are about bias, breaks, and the difference between an inaccurate forecast and an unexplained one. The first vignette below draws on the published record of retail forecasting practice; the second draws on the documented industry response to the 2020 disruption; the third is a composite of recurring professional patterns rather than a report about a single named organization.

10.14.1 Demand Planning, S&OP, and the Bullwhip

The first vignette is demand planning and sales-and-operations planning, the institutional home of forecasting in most product companies, and the place where a forecast most directly becomes money. The cadence is recognizable from this chapter's own decision context, run at industrial scale: a statistical baseline forecast is generated by item and location; planners and commercial teams adjust it in a consensus process; and the adjusted forecast drives purchase orders, production schedules, and inventory positions months ahead of the demand it predicts. Retail adds two complications the published research treats as central and this chapter has been building toward. Demand is hierarchical — the same forecast is wanted at chain, category, store, and SKU grain, and the accuracy that is achievable falls sharply as the grain gets finer, which is exactly why this chapter's deliverable declares a chain-level dollar envelope and refuses the unit plan. And promotions dominate the error: at store-SKU grain, a large share of the variance a retailer most needs to anticipate is generated by its own promotional calendar rather than by any underlying seasonal process (Fildes et al., 2022b).

Two failure patterns dominate the genre's scar tissue. The first is bias: consensus processes tilt optimistic, because targets, incentives, and enthusiasm all push one direction — which is why mature demand-planning organizations track signed error as seriously as absolute error, and audit whether human touches to the statistical forecast add accuracy at all, the practice the judgmental-adjustment evidence of Section 10.9 directly supports (Fildes et al., 2009). The second failure compounds down the supply chain: the bullwhip effect, in which modest variability or bias in demand signals amplifies as each tier — retailer, distributor, manufacturer — orders against its own forecast of the tier below, so a small forecast error at the shelf becomes a large swing at the factory (Lee et al., 1997). The practice lesson lands directly on this chapter's machinery: a forecast is an input to other people's decisions, its errors propagate with leverage, and the disciplines that look pedantic at the desk — bias tracking, preserved statistical baselines, documented adjustments — are, at supply-chain scale, the difference between a bad quarter and a warehouse of markdowns.

10.14.2 The Season the Models Never Saw

The second vignette is the structural break at civilization scale. In the spring of 2020, pandemic lockdowns delivered a break no training window contained, and forecasting systems across retail experienced it as a controlled experiment nobody wanted: models trained on years of stable seasonality met a world where grocery demand spiked, apparel and travel collapsed, and e-commerce absorbed years of channel shift in weeks. The instructive part is not that the forecasts were wrong — everything in Section 10.4 says they had to be — but what the field concluded afterward, and the postscript to the retail forecasting literature is unusually direct about it. Established forecasting approaches, tuned on stable history, degraded badly; the period exposed how thin the research base was

on forecasting under abrupt structural change and on instability in general; and the practical responses that helped were the ones that shortened the reach of history and widened the statement of uncertainty (Fildes et al., 2022a).

Read those responses against this chapter's repairs, because they are the same list applied under fire. Teams shortened training windows and down-weighted the broken period, which is the which-history-still-applies judgment of Section 10.10. They re-forecast more frequently, shrinking the horizon because expected error grows with it. They widened intervals and communicated ranges where they had shipped points, because the honest statement of uncertainty had become impossible to avoid. And they shifted weight from fitted history to structured judgment — documented adjustments carrying information no model possessed, which is Section 10.9 operating as the primary engine rather than the garnish. The durable lesson survived the emergency: the pandemic period now sits in every retailer's training history as a permanent monument to the principle that a model cannot announce its own obsolescence. The humans who knew the world had changed had to say so, in writing, and the organizations that could do that quickly were the ones whose forecasting practice already separated the statistical baseline from the judgmental layer.

10.14.3 Media Pacing as Weekly Micro-Forecasting

The third vignette is small, fast, and the one most students will live first; it is a composite of a practice pattern rather than a named case. Every performance marketing team runs a version of it: a monthly or quarterly budget, spend and revenue accruing daily, and a standing question — where will we land? — answered continuously from partial-period actuals. The naive arithmetic everyone starts with ("we are 40 percent through the month at 35 percent of budget, so we will underspend") is a forecast, and usually a bad one, because it ignores exactly the structure this chapter taught: day-of-week rhythm, so a month's spend is not linear in days; intra-month and moving-holiday seasonality, since auction prices and conversion shift toward month-end and toward the retail events whose dates move; and self-inflicted breaks — a creative refresh, a bid-strategy change, a competitor entering the auction. Teams that pace well run, in miniature, the full discipline: a seasonal-naive expectation of the period's shape, a short-horizon projection updated daily, explicit adjustments for the promotions and launches on the calendar, and a range rather than a point when the finance partner asks for the landing. The practice lesson is proportion: the horizon is days and the stakes are one budget line, but the habits — baseline first, calendar declared, adjustment documented, range communicated — are the identical habits the holiday plan needed, which makes pacing the training ground where a junior analyst can practice this chapter weekly with fast feedback.

10.14.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into one expectation, and it is this chapter's closing thread: businesses forgive inaccurate forecasts and punish unexplained ones. Every organization that plans has been missed by its forecasts — the veterans in the room have watched seasons collapse and surprises rally, and they know the future is not fully knowable. What they do not forgive is the miss that cannot be explained: the number with no method behind it, the point with no range around it, the adjustment nobody wrote down, the December surprise that turns out to have been an announced opening the forecast silently ignored, or a holiday the model was never told about. The analyst who ships the full deliverable — method, opponent, grade on a season that resembles the decision, ranges, calendar, override log, review date — will still be wrong, routinely, by roughly the amount the holdout promised. But she will be wrong in a way the organization can learn from, and her forecasts will be trusted more each season, because the trust was never really about accuracy. It was about accountability, and accountability is a document, not a talent.

10.15 Ethics, Forecast Accountability, and the Laundered Guess

The Business Analytics in Practice section ended on accountability as a professional survival skill; this section examines it as an ethical obligation, extending the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), causal language (Section 7.15), acting on predictions about people (Section 8.15), and fairness across groups (Section 9.15) — to what forecasting

adds: a number about the future that other people will stake resources, jobs, and judgment on, produced by a process only the analyst can see.

The chapter's ethics angle has a blunt name: the laundered guess. Laundering, in the financial sense, is passing something questionable through a process that makes it look clean, and a guess is laundered the same way: a number of uncertain provenance — a gut feeling, an unvalidated model, a target wearing analytical clothes — is passed through the apparatus of precision (decimals, a chart, a planning template) and emerges looking like knowledge. The vendor button of Section 10.1 launders structurally: a method nobody can inspect emits a number to the dollar, and the precision of the digits impersonates the reliability of the process. The consensus meeting launders socially: a forecast nudged upward by target pressure leaves the room wearing the statistical model's name. And the analyst launders personally in three specific ways this chapter has now named. She launders when she ships a point forecast without its range, because she possesses the holdout errors and has chosen to present a confidence the method does not have. She launders when she quotes a grade earned on a convenient window as though it applied to the decision's own season. And she launders when she reports a quantity her analytical frame cannot produce — a unit buy derived from a revenue forecast by assumptions nobody wrote down. In every case the mechanism is the same: uncertainty that someone knew about was stripped somewhere between the method and the meeting, and the audience downstream inherited a certainty nobody actually held.

Why is this an ethical matter rather than a technical one? Because of who bears the cost when the point forecast is treated as a promise. The planning lead who commits receipts against the laundered number owns the markdowns; the store teams whose staffing and bonus targets were set from it own the shortfall; the finance partner who guided the quarter on it owns the surprise. The analyst's exposure — an awkward January meeting — is routinely the smallest in the chain, which is precisely the asymmetry that makes the disclosure obligatory: the person most able to state the uncertainty is the person least exposed to the cost of concealing it. The interval is therefore not a statistical garnish but the ethical instrument of the trade — the sentence "between X and Y, eight times in ten, under the model, and here is what the model does not know" is the honest transfer of what the analyst knows to the people who will bear what she does not — and the guide's position is correspondingly plain: a point forecast delivered to a resource decision without its range, by an analyst who could have computed one, is a misrepresentation, however accurate the point later proves. Being right is not a defense; the audience was entitled to the uncertainty, and got confidence instead.

Accountability, the angle's other face, is what makes the obligation livable rather than paralyzing, and its instruments are the chapter's own. The forecast is signed — the analyst of record, per Chapter 1, extends through the forecast's assumptions and overrides, each owned by name with a review date. The uncertainty is communicated at the decision's grain — the interval with its width and period count, the horizon table, the scenario assumptions, and the statement of which instrument is the planning range at which horizon — so that leaning optimistic or conservative becomes the decision-maker's explicit, priced choice per Section 2.7 rather than the analyst's hidden one. The overrides are documented so that judgment is auditable, per Section 10.9, and reviewed against actuals so the organization learns which judgments to keep. The scope is declared, so that a dollar envelope is not silently consumed as a unit plan. And the forecast is kept separate, in writing, from targets and plans, because the measures-as-targets dynamic of Section 2.7 has a forecasting form with a long industrial history: the moment the forecast is graded on whether it matches the aspiration, it stops informing the aspiration, and the company is navigating by a compass wired to its own hull.

CONCEPT

The Interval Is the Ethics

Every discipline this chapter built converges on one exhibit: the honest range, stated in advance, carried into every artifact, graded after the fact. The interval discharges the analyst's deepest obligation in forecasting — to transfer, undamaged, her knowledge of her own ignorance to the people spending against it — and its absence is never neutral: someone downstream will fill the missing uncertainty with confidence, because that is what planning processes do to naked numbers.

> This chapter adds one clause the earlier draft of it lacked, and it is the clause an experienced forecaster will test you on. An interval is honest about what the model knows. It is silent about what the model was never told. On the sealed holiday holdout, the daily band achieved close to its nominal coverage while the season total landed near the top of its range, because the band priced noise faithfully and priced the missing calendar not at all. So the standing rule has five parts rather than four: no point without its range; no range without its method; no range without a statement of what it does not price; no override without its owner; no forecast without its review date. An analyst who holds that line will sometimes be the least popular voice in the planning meeting and will be, over seasons, the only one the meeting still believes.
>
> Source: Course concept developed for this guide, informed by Chatfield (1993) and Hyndman and Athanasopoulos (2021).

10.16 Chapter Summary

This chapter carried Part II's predictive discipline into time, and the move cost more than a change of index. The data structure came first: a time series is ordered, its order carries the information, and the exchangeability that let Chapters 8 and 9 shuffle customers freely died at the door — StyleCraft's daily revenue had to be manufactured from the fact table by a deliberate grain change, reconciled against its source on a complete calendar with the zero-fill rule stated rather than inherited from a library default, and read before it could be modeled. Reading meant decomposition: the series resolved into trend, seasonality, cycle, and noise, with the store-opening dates drawn onto the trend panel and the ten largest remainders joined to the campaign calendar against a printed chance expectation, so that "promotions explain the spikes" arrived as a comparison rather than an impression.

The chapter's organizing correction is a boundary. Calendar effects are knowable in advance, but only stable fixed-period patterns are carried automatically by seasonal memory; moving holidays, promotions, and openings must be supplied explicitly. A Holt-Winters model at seasonal_periods=7 tracks a weekly rhythm and nothing else seasonal, so StyleCraft's annual holiday structure — two observations of a 365-position pattern, one of them containing a store opening — is not fitted at all. It is carried instead by an explicit calendar layer estimated from the prior holiday and indexed on days from Thanksgiving rather than on calendar dates, by scenarios that vary that layer across the two years of evidence that exist, and by written overrides with owners and review dates. The toolkit was then built as a ladder with the opponent on the bottom rung: the naive and seasonal-naive rules extended Chapter 8's baselines into time and proved, per the M-competitions' standing humility, brutally hard to beat; moving averages smoothed the rhythm away for reading; and the exponential smoothing family — level, then Holt's trend, then a damped variant, then Holt-Winters' weekly seasonal effects — added tracked components one at a time, each purchasing power with flexibility.

Honest evaluation was rebuilt for ordered data in two layers rather than one. Random splits were convicted of leakage by design — the shuffled evaluation grades interpolation, a job no deployed forecast performs — and replaced by rolling origins inside the training data, which chose the method across seven origins and fifty-six-day horizons, and by one sealed holdout opened once for the frozen method. The holdout was chosen for a property most tutorials ignore: it had to resemble the decision, so it is a prior holiday season forecast from before it, and the result is the chapter's most useful number. The method that won on ordinary windows missed the holiday by roughly twice its selection score, systematically low, which is not a defect but the measured size of the calendar the model was never given. MAPE joined the dollar metrics as planning's shared currency and acquired a guard in code — denominators asserted, minimum printed beside every figure — after the miniature showed one storm-closed day converting a five-percent forecast into a headline of two hundred sixty. Error was reported by horizon across several origins rather than one path, with the number of errors behind each row printed, and the claim was stated at the strength the evidence supports: expected error and uncertainty grow with horizon, while realized error in any one sample need not rise at every step.

The forecast was then widened from a number into a distribution, and the widening carried two corrections. Weekly and season-total intervals are built by aggregating each simulated path first and taking quantiles second, because the sum of seven daily tenth percentiles is not the tenth percentile of the seven-day total — and the lab measured how wrong the shortcut is on real paths and on paths whose dependence had been shuffled away, to show that the size of the error cannot be known without the paths. And the band was qualified rather than trusted: it is conditional on the fitted parameters, it prices future innovation and horizon compounding, and it prices the missing calendar not at all — which the sealed holdout demonstrated by achieving near-nominal daily coverage while the season total landed near its upper bound. Coverage was reported with width, period count, and horizon detail, so that no one can buy a coverage figure by inflating a band.

The human hand was disciplined rather than banished. Judgmental adjustment earned its legitimacy exactly where it carries information the history cannot — the announced opening, the moving holiday, the drafted promotion — and paid for it with implementation as well as paperwork: a ramp profile rather than a flat step, because the prose said the stores ramp; equal weekday windows, excluded overlapping openings, excluded holiday-season openings, a promotion-day cap, stratification by store type, every surviving estimate printed, and a range carried forward instead of a decimal; plus a net-of-trend diagnostic that asks how much of the apparent step the model was already extrapolating, whose instability is itself the finding. Then the six-element written override, whose deepest payoff arrives in January when the preserved baseline lets the organization compute what each judgment contributed.

The verification theme ran through all of it: the labs forecast the known past across a programmatically selected opening, watched the well-fitted model and the free rule fail identically in direction, and closed both the level and the shape with a documented ramp. And the deliverable finally reached the decision without overclaiming: a Q4 scenario envelope with its assumption columns, a horizon caution stating that the graded horizon was fifty-six days and December sits three times further out, an open-to-buy envelope built from a declared identity whose every term varies by scenario, a media pacing shape at declared seven-day blocks, and a scope statement naming what a chain-level revenue forecast cannot produce — the unit buy, which is separate work at a separate grain. Twenty numbered cells carried the argument into executable form, and the failure modes, the AI supplement, the interpretation, the practice mirror, and the ethics section circled one insistence from different distances: a forecast is a probability statement with a method, an opponent, a grade on a season that resembles the decision, a range with a stated scope of what it prices, and a signature. Everything less is a guess, laundered.

One forward note redeems a pointer the smoothing section deferred: the moving averages built here return in Chapter 14 as visual analytics, computed live on the Tableau canvas, where their window arithmetic — learned here — is the difference between reading them and merely displaying them.

Looking ahead, the chapter's machinery answers what will happen: December's range, the season's shape, the opening's step. It is silent, by construction, on the question the planning meeting will ask next — what caused it, and what would happen if StyleCraft acted differently? The forecast can say the suburban stores' revenue will step up; it cannot say the expansion caused the step, or what December would have looked like without it, because everything in this chapter learned from a single unrepeated history in which StyleCraft did only one thing. The next chapter takes up the machinery that can answer such questions: experiments, A/B testing, and causal evidence — random assignment as the instrument that manufactures the comparison history never provides, the known true lift the dataset ships so the analysis can grade itself, and the evidence hierarchy that finally places everything Part II has built — descriptions, models, forecasts — on the ladder of what each one licenses a marketer to claim.

10.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: build and reconcile the series before modeling it, read the decomposition before choosing a method, name the opponent before fitting, separate method selection from the grade, choose a holdout that resembles the decision, report MAPE with its denominators and error by horizon across origins, aggregate intervals in the right order, document every override with its six elements, and declare the limits of the planning unit. They are organized into three groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a

subset; Assignment #4, Forecasting, draws on the homework sets of this chapter. In-class activities are prepared for discussion rather than submitted. Extensions are optional. Each exercise also carries its assignment label so that instructors can assign selectively.

10.17.1 Core Chapter Practice

Exercise 10.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- State what makes a time series a different data structure from the customer feature table, and name the discipline from Chapter 8 that survives the change and the one that does not.
- Distinguish seasonality from a cycle and both from a structural break, with one StyleCraft instance of each.
- Explain why history from before a structural break misleads a model after it, and why this is not a defect the model can be tuned out of.
- A colleague says that because Christmas and Black Friday are known in advance, a Holt-Winters model will carry them. Correct the claim precisely, distinguishing knowable from carried, and name the three instruments that carry the effects the model does not.
- Write the naive and seasonal-naive forecasting rules for a daily series with weekly rhythm, and explain why this guide states the December-repeats-December version only for monthly data.
- State what Holt's method adds to simple exponential smoothing, what the damped variant adds to Holt, and what Holt-Winters adds to both — and name the new failure each addition makes possible.
- Explain why a random train/test split flatters a forecasting model, using the word interpolation, and state the one-line check that catches the flattery.
- Explain why comparing four candidate methods on the final holdout spends the grade, and describe the two-layer design that fixes it.
- This chapter's sealed holdout is a prior holiday season rather than the last eight weeks of the file. Give the reason, and state what a late-spring holdout would and would not have established about a holiday plan.
- A holdout MAPE of 6 percent was computed on a series whose minimum value is $180 against typical values near $11,000. May the MAPE lead the deliverable? Explain, citing the standing rule, and describe what the chapter's mape helper would have done.
- Explain why the sum of seven daily 80% intervals is not the weekly 80% interval, and state the correct procedure in one sentence.
- A stakeholder reads an 80% prediction interval as "the analysts are only 80% sure of their work." Correct the reading in two sentences, including what it would mean for the interval to be wrong and what the interval does not price at all.

Exercise 10.2 Baselines and MAPE by Hand (Required Practice)

Using only the fourteen-day miniature of Code 10.1 (no code): rebuild both forecast rules day by day and verify every figure in the Verification Check — the naive rule's errors and its MAE of 1,300.00 and MAPE of 12.79 percent; the seasonal-naive rule's errors and its MAE of 514.29 and MAPE of 5.25 percent. Identify the two week-two days on which the naive rule's error is largest and state, in calendar language, what each miss has in common. Show that all seven seasonal-naive errors share a sign, compute their mean, and write one sentence on which row of Table 10.1 the bias reveals. Then rebuild the storm scenario: verify the MAE of 1,471.43 and the MAPE of 261.52 percent, show which single term produced the explosion, and state what value the storm Tuesday would have had to take for the chapter's guarded mape helper to refuse to compute at all. Close with the two-sentence honest summary of the storm week that the standing denominator rule requires.

Exercise 10.3 Read the Calendar (Required Practice)

For each series below, list the calendar effects and structural breaks a forecaster should declare before fitting, classify each as living in the training window, the forecast window, or both, and state — per Section 10.4's obligation — how each forecast-window item should be discharged: by a fitted seasonal period, by an explicit

calendar layer, by a documented override, by a stated scenario assumption, or by scope exclusion. Where you answer "fitted seasonal period," you must also state how many repetitions of the pattern the history contains.

- StyleCraft's daily chain revenue, forecast from July 2026 through the end of December 2026, given the stores table, the drafted holiday promotion calendar, and a board deck listing two further openings under consideration for October.
- A single Wave-4 resort store's daily revenue, opened five months ago, forecast eight weeks ahead.
- The email channel's daily attributed revenue, where sends occur three days a week and revenue on non-send days is frequently near zero.
- Weekly chain revenue for the four weeks around Thanksgiving, forecast one year ahead, where the analyst proposes to use last year's same-week values.

Exercise 10.4 Spot the Failure (Required Practice)

Each scenario below contains at least one failure from this chapter, or none. Name it, cite the section or table row, and state the repair.

- An analyst evaluates a daily revenue model with five-fold cross-validation, reports a MAPE of 2.1 percent, and notes the model "performs far better than the seasonal-naive rule's 7 percent."
- A pipeline's feature list includes a centered 14-day moving average of revenue.
- An analyst compares naive, seasonal-naive, and Holt-Winters on the final eight weeks of the series, keeps Holt-Winters because it wins, and reports its error on those eight weeks as the model's out-of-sample grade.
- A forecast for October–December is fitted on the full two-year history, and the analyst notes with satisfaction that the model has "learned the holiday season" from granular day-of-year seasonal effects.
- A team forecasts across an announced flagship opening using Holt-Winters, and when the forecast undershoots, concludes the smoothing parameters need retuning.
- A weekly forecast table reports 80% bounds obtained by summing the daily 80% bounds within each week. The analyst notes that the difference "is only a few percent anyway."
- An analyst reports the eight-week forecast as a weekly table of point values with a footnote that "actual results may vary," and separately keeps a spreadsheet of the intervals "to avoid confusing the audience."
- A planning lead fits the forecast, finds it 9 percent below the CEO's announced growth target, and raises the seasonal effects until the gap closes, noting the December numbers "looked low anyway."
- A deck reports a chain revenue forecast of $1.7 million for Q4 and, on the next slide, a recommended purchase quantity of 34,000 units. No other assumption appears anywhere in the deck.

Exercise 10.5 The Full Forecasting Deliverable (Homework Submission)

Complete Labs 10.1 and 10.2 on the certified files and assemble the planning deliverable per Section 10.13: the frame-and-calendar page with the reconciliation certificate and the three declared windows with their different rights; the rolling-origin leaderboard for all four candidates with MAPE, minimum denominators, MAE, the week-eight column, the bias column, and the margin over the opponent in dollars per day; the sealed-holdout grade, opened once, for the frozen method and the frozen method plus the calendar layer, with signed error and season-total miss; the horizon table with its error counts and the December-distance sentence; the inflection exhibit with both methods' bias across the programmatically selected opening and the ramp override's measured contribution to level and to shape; the override log with all six elements per entry, including the range and the net-of-trend figure in the evidence field; the interval paragraph with coverage reported beside width and period count, the weekly and season-total bands built path-first, and the sentence stating what the band does not price; the scenario table with its assumption columns and the horizon caution; and the planning translation with the open-to-buy envelope, its sensitivity to the assumption you identified in Code 10.21's Verification Check, the asymmetry of Section 2.7 argued explicitly, the media pacing shape, the scope statement, and the re-forecast calendar. Submissions are graded on the calendar page, the separation of selection from grade, the override log, the interval communication, and the scope statement as heavily as on the models.

Exercise 10.6 AI Forecast Audit (Homework Submission)

Give an AI assistant the raw marketing_daily file and this deliberately loose prompt: "Forecast our daily revenue for the next eight weeks — maximize accuracy." Before reading the response, write your predicted leaderboard band per the AI in Practice box: the seasonal-naive MAPE you expect (compute it yourself) and the band an honest model should land in. Then audit the response in writing with Table 8.5's five points followed by Table 10.5's four: locate the split and run the date-order check; determine whether method selection and the reported grade happened on the same data; determine whether any feature or smoother consumed post-cutoff data; reconstruct what the pipeline assumed about openings, promotions, and the holiday; check every metric's denominators and every forecast for its missing interval, including whether any weekly or total interval was built by summing daily bounds; and compare the headline claim against your predicted band, classifying any excess as pipeline, not prophecy, until traced. Re-prompt with the full frame and compare leaderboards. Document both exchanges per the AI-use documentation template in Appendix D, and conclude with two sentences on which supplement point caught the most serious problem.

10.17.2 In-Class Activities

Exercise 10.7 The Override Meeting (In-Class Discussion)

Your team will rehearse the planning meeting's hardest conversation: which judgments enter the forecast. Prepare: the statistical baseline forecast and its two grades, the rolling-origin score and the sealed-holiday score, with a one-sentence explanation of why they differ; a proposed override for the fall's announced openings, sized per Codes 10.13 through 10.15, with its six elements complete and its range and net-of-trend figure in the evidence field; a proposed calendar layer for the holiday, with the contamination in its single prior observation disclosed; the merchant's thirty-percent gifting thesis, translated into the strongest legitimate form it can take under Section 10.9 — a named scenario with an argued size, prepared from occasionwear mix and the suburban stores' basket profile, per the certified findings of Chapter 5; and a written position on which of the three circulating forecasts from Section 10.1 contains information the final forecast should keep, and in what form. Half the class prepares the merchant's side seriously: her number cannot be graded, but her knowledge is real, and the strongest version of her case is what the override discipline must be able to absorb without breaking.

Exercise 10.8 Find the Flaw in the Forecast (In-Class Discussion)

Each scenario below contains at least one flaw from this chapter. Name it, cite the section, and state the repair.

- A vendor's forecasting module reports 97 percent accuracy. Its documentation does not state the evaluation design, and its demo notebook calls a generic train/test split on the daily rows.
- A forecast deck leads with a single MAPE for the eight-week horizon. The commitment concerns weeks seven and eight.
- A team validates its holiday forecasting method on the eight weeks ending in June, then uses it to set November and December receipts.
- A planning team's December forecast has come in optimistic four years running. Each year's miss was explained by a different one-off; no one has computed the four-year mean signed error.
- An analyst, told the openings would bias her forecast low, raises every week of the forecast by 6 percent, mentions it in the meeting, and moves on. The next analyst inherits the model and cannot reproduce the published numbers.
- A forecast's 80% intervals captured the actuals in 94 percent of holdout periods, and the team celebrates the intervals as "extra safe." No interval width is reported anywhere in the deck.
- An 80% interval on the season total is reported for a forecast issued six months ahead, with no statement of the horizon at which the method was graded.
- A dashboard recomputes the season forecast nightly, and the number the CFO quotes moves daily by amounts that dwarf the model's day-over-day information gain. Planning has begun averaging the last five days' numbers by hand.

10.17.3 Extensions

Exercise 10.9 Build the Horizon Curve Two Ways (Optional)

The horizon table of Code 10.10 averages errors by week-ahead across seven forecast origins. Rebuild it two ways and compare. First, from a single origin — take the last origin only and read its eight weekly blocks as a horizon curve, which is what the draft of this chapter did. Second, from the seven origins as the chapter now does. Report both tables side by side, with the number of errors behind each figure, and then answer three questions in writing. Where do the two curves disagree most, and what calendar event from Codes 10.4 and 10.5 sits underneath that week in the single-origin version? What does the standard-deviation column in the multi-origin table say about how much of the single-origin curve's shape was signal? And how many origins would you want before you were willing to publish a horizon curve as a claim about the method rather than a description of one two-month stretch — state a number and defend it. As an extension of the extension, vary the origin spacing from 28 days to 14 and report what changes and what does not, noting that overlapping forecast windows share observations and therefore share error.

10.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Calendar effect. A systematic influence the calendar exerts on a business series — day-of-week rhythm, fixed-date holidays, moving retail events, paydays, self-imposed merchandising rhythms. All are knowable in advance; only stable fixed-period patterns are carried automatically by seasonal memory, and the rest must be supplied explicitly (adapted from Hyndman & Athanasopoulos, 2021).

Cycle. A recurring rise and fall in a time series without a fixed, calendar-locked period — economic or fashion waves whose timing cannot be read from a calendar, distinguishing them from seasonality (adapted from Hyndman & Athanasopoulos, 2021).

Decomposition. The separation of an observed time series into estimated trend, seasonal, and remainder components under an assumed structure, most commonly additive; used in this guide as a reading instrument that shows which stories a series contains and how large each is, at one declared period (adapted from Cleveland et al., 1990; Hyndman & Athanasopoulos, 2021).

Exponential smoothing. A family of forecasting methods that maintain running, recency-weighted estimates of a series' components — level alone in simple smoothing, level and trend in Holt's method, level, trend, and season in Holt-Winters — updating each by a fitted smoothing parameter as observations arrive (adapted from Holt, 2004; Gardner, 2006).

Forecast horizon. The span of future periods a forecast claims, in the series' own grain. Because uncertainty compounds with each step ahead, forecast uncertainty and expected error generally increase with horizon, although realized error need not rise monotonically in any one evaluation period; honest evaluation reports accuracy by horizon, estimated across several forecast origins, with the number of errors behind each figure stated (adapted from Hyndman & Athanasopoulos, 2021; Tashman, 2000).

Holiday calendar layer. This guide's name for an explicit, analyst-supplied adjustment that carries annual and moving-holiday structure a fitted seasonal period cannot: a ratio of actual to statistical forecast measured across a prior holiday season and indexed by days from the moving anchor date rather than by calendar date. Estimated from one or two observations, it is an assumption with a stated range and a review date, not a fitted component (course concept developed for this guide, informed by Hyndman & Athanasopoulos, 2021).

Holt-Winters method. The seasonal member of the exponential smoothing family: smoothed level, smoothed trend, and smoothed seasonal effects at a declared period, forecast forward as a trended level with the seasonal pattern reapplied. A model fitted at period 7 carries a weekly pattern and no other (adapted from Winters, 1960; Gardner, 2006).

Judgmental adjustment. A deliberate human change to a statistical forecast intended to incorporate information the fitted method could not access; legitimate when carrying genuine event knowledge and documented as a written

override, and reliably accuracy-destroying when small, frequent, and optimistic — with upward adjustments substantially less reliable than downward ones (adapted from Fildes et al., 2009).

MAPE (mean absolute percentage error). The average of absolute forecast errors expressed as percentages of the actual values; scale-free and planning's shared currency, but undefined at zero actuals and explosive near them, so it is computed only after its denominators are asserted, reported with its minimum denominator, and disqualified for series that approach zero (adapted from Hyndman & Koehler, 2006).

Moving average. The replacement of each observation with the mean of a fixed window of surrounding observations, trailing or centered; a window equal to the seasonal period removes the seasonal pattern, making the moving average both a trend-extraction instrument and a deliberate act of information removal. Inside a forecasting pipeline it must be trailing only (adapted from Hyndman & Athanasopoulos, 2021).

Naive forecast. The rule that every future value equals the most recent observed value — the last-value baseline of Section 8.7 at the series' own grain, extended flat across the horizon (adapted from Hyndman & Athanasopoulos, 2021).

Open-to-buy envelope. This guide's name for the planning unit a chain-level revenue forecast can honestly produce: planned receipts at retail, computed as planned sales plus planned ending inventory plus planned markdowns minus inventory on hand, converted to dollars of buying authority by a declared cost complement. Every term is either forecast or declared and varies by scenario, and the envelope is a dollar figure rather than a unit plan (course concept developed for this guide).

Prediction interval. A range around a point forecast derived from the forecast distribution the fitted model implies: under the model and its assumptions, an 80% interval is constructed so that comparable intervals would contain their future observations about 80% of the time. Its width grows with horizon, it is gradable by realized coverage reported beside width and period count, and it is conditional on the fitted parameters — silent about structure the model was never given (adapted from Chatfield, 1993; Hyndman & Athanasopoulos, 2021).

Rolling-origin evaluation. Estimation of out-of-sample forecast accuracy by repeating a fit-and-forecast exercise from several successively later forecast origins, so that every training set precedes its own test observations and accuracy can be reported separately by horizon; the time series counterpart of cross-validation, performed inside the training data so a final holdout can stay sealed (adapted from Tashman, 2000; Hyndman & Athanasopoulos, 2021).

Sealed holdout. A contiguous block of the known past, withheld from every selection decision and consulted once for the frozen method. In forecasting it carries an additional obligation the customer-grain version did not: it should resemble the decision, in season and in horizon, because a method graded on an unlike window has not been tested on the risk the decision runs (course concept developed for this guide, informed by Tashman, 2000).

Seasonal-naive forecast. The rule that every future value equals the most recent observed value from the same position in the seasonal cycle at a declared period — this Saturday forecast by last Saturday on daily data at period 7, this December by last December on monthly data at period 12 — reproducing one cycle of the pattern forward at zero fitted parameters; this guide's declared opponent for forecasting models (adapted from Hyndman & Athanasopoulos, 2021).

Seasonality. A repeating pattern in a time series with a fixed and known period, recurring at a constant number of observations apart — days of the week, months of the year — and therefore carryable by seasonal memory. Moving holidays are calendar effects but not seasonality in this sense (adapted from Hyndman & Athanasopoulos, 2021).

Structural break. A point in time at which the process generating a series changes — a persistent shift in level, trend, or seasonal pattern — after which pre-break history describes a regime that no longer exists; in marketing, frequently self-inflicted and announced in advance (adapted from Hyndman & Athanasopoulos, 2021).

Time series. A sequence of observations of the same quantity at successive, usually regular, points in time, in which the temporal ordering is part of the data's meaning and must be preserved because observations may exhibit serial dependence, trend, seasonality, or structural change (adapted from Hyndman & Athanasopoulos, 2021).

Trend. The long-run direction of a time series — the smooth underlying path once short-run fluctuation is set aside; in StyleCraft's series, organic growth compounded by the expansion's stair-step contributions (adapted from Hyndman & Athanasopoulos, 2021).

10.19 Further Readings

Students who want additional background may begin with the following readings. The applied treatments are listed first, evidence and methods second, software last.

- Hyndman and Athanasopoulos (2021) for the standard modern textbook of applied forecasting — free online, code-first, and the source this chapter's definitions lean on most. The chapters on decomposition, exponential smoothing, and evaluation are the direct expansions of Sections 10.3, 10.6, and 10.7; the section on time series cross-validation is the formal treatment of the rolling origins this chapter's Code 10.8 implements; the section on weekly, daily, and sub-daily data is where the multiple-seasonality and moving-holiday problem of Section 10.4 is developed properly, including the harmonic-regression machinery this chapter deliberately leaves out; and the section on distributional accuracy is where coverage-plus-sharpness evaluation goes beyond what Section 10.8 does.
- Fildes et al. (2022b) for the field's own survey of retail forecasting research and practice — hierarchical demand, the dominance of promotions at fine grain, and the gap between what the literature studies and what retailers do; the direct expansion of Section 10.14.1 and the published support for this chapter's refusal to promise a unit plan from a chain-level series. Read it with its postscript, Fildes et al. (2022a), which is the short and pointed account of what the 2020 disruption did to established methods and why structural instability remains the field's weakest area.
- Makridakis et al. (2020) for the M4 competition's results — one hundred thousand series, sixty-one methods, and the empirical case for baseline humility that Section 10.5 built its opponent on.
- Tashman (2000) for the methodological treatment of out-of-sample testing: fixed versus rolling origins, why multiple origins are needed before a horizon curve means anything, and the design vocabulary behind Table 10.3.
- Hyndman and Koehler (2006) for the definitive survey of forecast accuracy measures, including the full indictment of MAPE's denominator behavior and the scale-free alternatives this guide leaves for later coursework.
- Fildes et al. (2009) for the large-scale field evidence on judgmental adjustments in real forecasting operations — which adjustments help, which reliably hurt, why upward revisions are the least trustworthy, and why the written-override discipline of Section 10.9 is what separates them.
- Gardner (2006) for the authoritative survey of exponential smoothing's state of the art, including the damped-trend evidence behind this chapter's fourth candidate — the extended companion to Section 10.6, one level of formality above this chapter.
- Lee et al. (1997) for the bullwhip effect — the classic analysis of how forecast errors and ordering behavior amplify up a supply chain, and the reason Section 10.14.1 treats forecast bias as an operations problem rather than a statistics problem.
- The statsmodels documentation for the three functions this chapter's labs depend on — seasonal_decompose, ExponentialSmoothing, and HoltWintersResults.simulate (statsmodels developers, 2026a, 2026b, 2026c) — and the development release notes for the forthcoming 0.15.0 release (statsmodels developers, 2026d) for the simulation seed argument's planned rename from random_state to rng, which is the reason Section 10.12 pins a stable version and states it. The library itself is Seabold and Perktold (2010).

10.20 References

Chatfield, C. (1993). Calculating interval forecasts. *Journal of Business & Economic Statistics, 11*(2), 121–135. https://doi.org/10.1080/07350015.1993.10509938

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics, 6*(1), 3–73.

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2009). Effective forecasting and judgmental adjustments: An empirical evaluation and strategies for improvement in supply-chain planning. *International Journal of Forecasting, 25*(1), 3–23. https://doi.org/10.1016/j.ijforecast.2008.11.010

Fildes, R., Kolassa, S., & Ma, S. (2022a). Post-script—Retail forecasting: Research and practice. *International Journal of Forecasting, 38*(4), 1319–1324. https://doi.org/10.1016/j.ijforecast.2021.09.012

Fildes, R., Ma, S., & Kolassa, S. (2022b). Retail forecasting: Research and practice. *International Journal of Forecasting, 38*(4), 1283–1318. https://doi.org/10.1016/j.ijforecast.2019.06.004

Gardner, E. S., Jr. (2006). Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting, 22*(4), 637–666. https://doi.org/10.1016/j.ijforecast.2006.03.005

Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting, 20*(1), 5–10. https://doi.org/10.1016/j.ijforecast.2003.09.015

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts. https://otexts.com/fpp3/

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Lee, H. L., Padmanabhan, V., & Whang, S. (1997). Information distortion in a supply chain: The bullwhip effect. *Management Science, 43*(4), 546–558. https://doi.org/10.1287/mnsc.43.4.546

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting, 36*(1), 54–74. https://doi.org/10.1016/j.ijforecast.2019.04.014

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference* (pp. 92–96). https://doi.org/10.25080/Majora-92bf1922-011

statsmodels developers. (2026a). *statsmodels.tsa.seasonal.seasonal_decompose* [Software documentation]. https://www.statsmodels.org/stable/generated/statsmodels.tsa.seasonal.seasonal_decompose.html

statsmodels developers. (2026b). *statsmodels.tsa.holtwinters.ExponentialSmoothing* [Software documentation]. https://www.statsmodels.org/stable/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html

statsmodels developers. (2026c). *statsmodels.tsa.holtwinters.HoltWintersResults.simulate* [Software documentation]. https://www.statsmodels.org/stable/generated/statsmodels.tsa.holtwinters.HoltWintersResults.simulate.html

statsmodels developers. (2026d). *Development release notes for the forthcoming 0.15.0 release* [Software documentation]. https://www.statsmodels.org/dev/release/version0.15.0.html

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: An analysis and review. *International Journal of Forecasting, 16*(4), 437–450. https://doi.org/10.1016/S0169-2070(00)00065-0

Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science, 6*(3), 324–342. https://doi.org/10.1287/mnsc.6.3.324

Chapter 11

Experiments, A/B Testing, and Causal Evidence

From Prediction to Cause

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter builds the causal branch of marketing evidence, with StyleCraft's Drop Alert rollout decision as its running problem. It establishes what causal claims require: the counterfactual, potential outcomes at intuition level, and the fundamental problem that only one of a customer's two possible outcomes is ever observed. Selection bias is then named as the reason observational comparisons flatter targeted campaigns, and decomposed arithmetically on a record where the treated group's composition accounts for two-thirds of the apparent effect. Random assignment follows, with control design, the unit of randomization, preregistration as the experimental form of the analytic specification, statistical power, intention-to-treat analysis, and the checks that grade the assignment before the analysis grades the campaign. The chapter then measures lift with its interval, prices it against a break-even line computed from the program's own dose and economics, catalogs five testing failures, bounds quasi-experimental inference, and closes on the evidence hierarchy.

KEYWORDS

causal inference; counterfactual; selection bias; random assignment; A/B testing; preregistration; statistical power; lift; experimentation; difference-in-differences

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Experiments, A/B testing, and causal evidence. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 11, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, NumPy, Matplotlib, and scikit-learn are sponsored or affiliated projects of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. statsmodels is a community-developed project distributed under the modified BSD license. ChatGPT is a product of OpenAI, Claude of Anthropic, Gemini and NotebookLM of Google LLC, and GitHub Copilot of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: Applied Business Analytics companion repository on GitHub

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and

analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

- Explain why marketing decisions are causal questions, state the fundamental problem of causal inference in the potential-outcomes frame at intuition level, and identify the missing counterfactual behind any claim that a campaign "worked."
- Define selection bias, explain why observational comparisons systematically flatter well-targeted marketing programs, state what determines the direction of the bias, and decompose an observed group difference into a treatment component and a composition component on a worked example.
- Explain what random assignment buys — comparability in expectation on all characteristics, measured and unmeasured — state the additional conditions an unbiased effect estimate requires, and design a treatment and control comparison for a stated marketing decision.
- Choose a unit of randomization appropriate to a marketing intervention, state the contamination risk that follows from choosing a unit smaller than the unit the treatment actually reaches, and explain why effect estimates and standard errors must respect the assignment grain.
- Write a preregistration for a marketing experiment — decision, primary metric, arms, treatment dose, unit of randomization, sample size, duration, decision rule, and stopping rule — as the experimental form of the analytic specification of Section 2.5, and confirm that every declared guardrail is computable from the certified data.
- Explain statistical power and the relationship between detectable effect size and required sample size, and compute the sample size implied by a stated minimum detectable lift, baseline rate, significance level, and power target.
- State the intention-to-treat principle, build an assignment-flow table from randomization through outcome observation, and explain why post-assignment filtering recreates the selection the design was built to remove.
- Interpret a covariate balance check and a sham-split analysis-path check correctly, distinguish both from a genuine end-to-end A/A test, and state what each failure mode of the assignment machinery would look like in their output.
- Compute absolute and relative lift for a conversion experiment, construct a confidence interval for a difference in proportions and a separate interval for a ratio of proportions, and report the result in percentage points, in relative terms, and in dollars.
- Distinguish statistical from practical significance, define a program's treatment dose, compute a break-even lift from that dose and the program's contribution, and state a rollout decision rule that compares the full interval against that line under stated assumptions.
- Recognize and prevent the five standing testing mistakes — peeking, multiple comparisons, mid-test changes, novelty effects, and contamination — and identify each from the symptoms it leaves in a test report.
- Describe pre/post with a comparison group and difference-in-differences at concept level, state the parallel-trends assumption these designs require, and explain what they can and cannot establish when randomization is unavailable.
- Place a piece of marketing evidence on the evidence hierarchy, state the strongest sentence each rung licenses, and state the execution-quality condition the hierarchy assumes.

- Audit an AI-drafted experiment design and analysis with the five-point audit of Section 8.11 plus the experimentation supplement — registration precedes analysis, assignment integrity verified, inference honesty checked — predicting the estimate before running it, and documenting the audit per Appendix D.
- Assemble a rollout recommendation a chief marketing officer can act on and a finance partner can challenge: registration, assignment certificate, lift with interval in decision units, break-even comparison with its sensitivities, extrapolation assumptions named, and the guardrails under which the rollout would be reversed.

Chapter 10 ended with a forecast that could be graded. The buy plan's number arrived with a method, an opponent it had beaten, a sealed-holdout grade in dollars and percentages, a prediction interval that stated its own ignorance, and an override log that made the analyst's private knowledge of the calendar into a public, reviewable input. That deliverable answers one class of question completely: what will happen if StyleCraft keeps doing what it is doing. It answers no version whatever of the question the same meeting asks thirty seconds later — what would happen if we did something different — and the reason is structural rather than technical. Every method in Part II learned from a single history in which StyleCraft did exactly one thing. The forecast can report that suburban revenue stepped up in the weeks after the Greenwich opening; it cannot report what those weeks would have contained had the store never opened, because that version of the fall does not exist in any file and never will. The gap between the two statements is not a gap in the data's size or the model's sophistication. It is a gap in the kind of evidence that observation can produce at all, and it is the reason this chapter exists. Marketing decisions are actions — send it, don't send it, spend here, spend there — and an action is a claim about a world that has not happened yet. This chapter builds the only instrument in the guide that manufactures evidence about such worlds on purpose: the controlled experiment, in which the analyst does not wait for a comparison to appear in the data but creates one, by deciding at random who gets the treatment and who does not. The chapter's arithmetic is the simplest in Part II — two proportions and their difference — and its discipline is the strictest, because an experiment's credibility is spent almost entirely in the choices made before any number is computed, and can be destroyed afterward by a single unplanned look at the results.

CONCEPT

What This Chapter Is Really About

Every chapter of Part II has produced a number that describes the world StyleCraft already lives in. The segments describe who its customers are; the regression describes which characteristics accompany higher spend; the churn model describes who is likely to leave; the forecast describes where the series is heading. Not one of those numbers, however carefully earned, answers the question that actually precedes a marketing budget: if we act, what changes? The obstacle is not that the models are weak. It is that the data records a world in which StyleCraft's choices were made by StyleCraft — targeted at the customers judged most likely to respond, timed to the seasons with the most demand, aimed at the stores already performing — so that every apparent effect in the record arrives fused to the reasons it was applied where it was. Untangling the two by thinking harder is impossible in principle, because the comparison the question needs is a version of the same customers, in the same season, who did not receive the campaign, and that version was never recorded.

> The experiment's contribution is to stop trying to untangle and start manufacturing: if a chance mechanism decides who is treated, then the treated and untreated groups differ, on average, in exactly one respect — the treatment — and their difference in outcomes estimates the effect with nothing else fused to it. That single structural move is the whole chapter, and everything else in it exists to protect the move from being undone. The preregistration protects it from the analyst's own hindsight. The integrity checks protect it from a broken randomizer. The intention-to-treat rule protects it from a delivery failure quietly rebuilding the selection. The power calculation protects it from a test too small to see the effect it was built to find. The stopping rule protects it from the peeking that manufactures winners out of noise. The break-even line protects the result from being read as a business case it does not support. And the evidence hierarchy protects the rest of the guide's output from being described in words only an experiment has earned. An experiment is the most trustworthy instrument in this book, and the most fragile.
>
> Source: Course concept developed for this guide, informed by Kohavi et al. (2020) and Imbens and Rubin (2015).

11.1 Marketing Decision Context: Did the Campaign Work, or Did the Customers?

The Comeback Edit ranked customers by risk; the buy plan committed inventory against a season. The decision that opens this chapter spends no money on customers at all until someone settles an argument about a number that has already been computed, twice, by two people who do not trust each other.

The number belongs to the Drop Alert. StyleCraft's merchandising rhythm is the weekly drop — a Thursday release of new styles, announced across email, social, and the app — and eight months ago the CRM team launched a variation on it: an SMS and app-push notification giving recipients a twelve-hour early-access window to the week's drop before the general announcement. One alert, one drop. The program costs roughly twelve cents per enrolled customer per alert once messaging fees and the operations of staging an early window are counted, and it has been running on the customers the CRM team judged most likely to use it — the opted-in, the app-installed, the loyalty-enrolled. The numbers have looked extraordinary. Customers who received the Drop Alert converted at 10.1 percent in the two weeks following a drop; customers who did not converted at 4.8 percent. The difference is 5.3 percentage points, which the CRM team's deck renders, correctly as arithmetic, as a lift of more than 110 percent.

On the strength of that deck, the chief marketing officer has proposed a reallocation for the fourth quarter: four hundred thousand dollars moved out of paid social and into an expansion of the Drop Alert to the full SMS-eligible base, beginning with the November drop calendar. The logic is not foolish. A program that doubles conversion at twelve cents a message is, if the number means what it appears to mean, the best-performing marketing StyleCraft owns, and the holiday quarter is precisely when it should be scaled. The CRM calendar for November and December locks in three weeks, because creative, merchandising, and messaging operations all schedule against it, which converts an interesting analytical question into a dated one.

The finance partner has blocked the reallocation, and her objection is not about the arithmetic. It is that the two groups were never comparable. The customers who received the Drop Alert were selected — deliberately, and for good operational reasons — because they were the most engaged customers StyleCraft has: app users, SMS opt-ins, the loyalty tiers Chapter 6's segmentation named the Urban Loyal Core. The customers who did not receive it are disproportionately the Suburban Occasion and New/At-Risk populations, who would convert at a fraction of the Core's rate if StyleCraft sent them nothing at all. Her question is the one this chapter answers: how much of the 5.3 points is the Drop Alert, and how much is the difference between the two kinds of customer who happened to land on either side of the comparison? She will fund the expansion, at scale, if someone can show her a number that is not contaminated by who was chosen.

So the VP of Marketing has borrowed the analyst — still you — and the commission is different in kind from every commission in Part II. Nothing in the existing data will settle the argument, because it records exactly one world: the one in which StyleCraft sent the Drop Alert to its best customers. What settles it already exists. Six weeks ago, anticipating this fight, the CRM team ran a proper test: ten thousand eligible customers were listed, a seeded draw selected exactly five thousand to receive one Drop Alert for the drop in their assignment week, the remaining five thousand received the standard announcement for the same drop, and conversion within the following fourteen days was declared in advance as the deciding metric. That file is the ab_test table, and what this chapter builds from it is the campaign's causal effect on conversion — a lift with an interval, priced at a stated dose against the break-even line that makes the rollout worth funding, with an honest account of what the experiment cannot say about a rollout differing from the test in population, duration, and dose.

Two features of the commission shape everything after. The first is that the answer is knowable in advance, and this is deliberate: the design note published with the course data dictionary states the effect the generator planted, a lift of 1.8 percentage points — large enough to be detectable at the planned sample size, small enough that a sloppy analysis will miss it or exaggerate it. That is the chapter's verification theme, and the analysis therefore grades itself. The second is that the observational number is not a lie. Both rates are real and both were computed correctly. The 5.3-point difference is a true fact about StyleCraft's records and a false answer to the CMO's question, and holding both in mind at once is the discipline this chapter installs.

The receipts lock in three weeks. Sections 11.2 and 11.3 explain why the true fact is the wrong evidence. Sections 11.4 and 11.5 build the design that produces the right evidence, size it, and audit it. Sections 11.6 and 11.7 measure the effect and price it against the program's dose. Section 11.8 catalogs the ways a sound measurement is destroyed after the fact. Sections 11.9 and 11.10 handle the decisions no chance mechanism can reach and place all of Part II's output on the ladder of claims. Section 11.11 turns to the assistant. The labs in Section 11.12 recover the planted lift and manufacture a false winner on purpose, and Sections 11.13 through 11.15 rehearse the meeting where an interval becomes a rollout.

11.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The CRM team's 5.3-point difference and the experiment's answer will disagree by a factor of roughly three. Before reading further, write one sentence naming the mechanism that produces the gap, and one sentence explaining why adding control variables to a regression on the observational data would not reliably close it.
- The finance partner asked for "a number that is not contaminated by who was chosen." State, in your own words, what property of the experiment's assignment procedure delivers exactly that, and what the property guarantees about characteristics of customers that StyleCraft has never measured.
- The experiment measured conversion over fourteen days among SMS-eligible customers during a non-holiday period, at a dose of one alert. The rollout would run for a quarter, across the holiday calendar, at a higher alert frequency. List three reasons the experimental estimate might not transfer, and note which of them this chapter names.
- The Drop Alert costs twelve cents per customer per alert, and the test delivered one alert per treated customer. Before Section 11.7 tells you, work out what additional piece of information you would need to decide whether a 1.8-point lift is worth having, and write the arithmetic you would perform once you had it.
- Suppose the experiment had been run at two alerts per treated customer instead of one, with everything else unchanged. State what that would do to the cost per assigned customer, to the break-even lift, and — this is the part worth thinking about — to the recommendation the same measured interval would support.

11.2 The Fundamental Problem of Causal Inference

The opening case turns on a question that sounds empirical and is, in a specific and permanent sense, unanswerable as posed: did the Drop Alert cause the conversion? This section explains why the question cannot be answered about

any individual customer, ever, by anyone, with any amount of data — and why that impossibility, properly understood, is the design brief for everything the rest of the chapter builds.

Start with one customer. C004471 received the Drop Alert on a Thursday in September and bought a dress the following Tuesday. Did the Drop Alert cause the purchase? The honest form of the question is comparative: what would C004471 have done if she had not received it? There are exactly two possibilities to compare — the outcome under treatment and the outcome under no treatment — and this pair is what the modern causal literature calls her potential outcomes (Rubin, 1974; Holland, 1986). Her causal effect is the difference between the two. And the difficulty, which no technology will remove, is that the world assigned her one of them. She received the alert; she bought; the version of C004471 who did not receive the alert is not in the file, not recoverable from the file, and does not exist. Her second potential outcome is not missing in the sense that Chapter 4's defect catalog uses the word, where a value was recorded badly or lost in transit. It is missing in the sense that it never happened.

DEFINITION

Counterfactual and Potential Outcomes

For a unit that could be exposed to a treatment or not, the potential outcomes are the pair of results that would occur under each exposure: the outcome under treatment and the outcome under no treatment. The unit's causal effect is the difference between the two. The counterfactual is whichever of the pair did not occur — the outcome in the world that was not realized. Because a unit is exposed either to the treatment or to its absence but never to both, exactly one potential outcome is observed and the other is permanently unavailable, which is why causal effects are estimated by comparison across units rather than measured within them.

Source: Adapted from Rubin (1974) and Holland (1986).

In other words, every causal claim in marketing is a claim about something that did not happen. "The campaign lifted conversion by 1.8 points" is shorthand for "conversion among these customers was 1.8 points higher than it would have been in a version of the same fortnight in which they received nothing," and the second half of that sentence is a description of a world nobody observed. Holland (1986) gave the situation its standing name — the fundamental problem of causal inference — and the name is worth taking literally. It is fundamental in that it is a feature of the logic of causation rather than a limitation of current measurement; it is a problem in that it makes the quantity of interest structurally unobservable; and it belongs to inference, not to data collection, because the response is not to collect the missing outcome but to construct a defensible estimate of it.

DEFINITION

Causal Inference

Causal inference is the estimation of what an outcome would have been under an alternative exposure — the counterfactual — using observed data plus assumptions or design features that make some observed group a credible stand-in for the unobserved one. Its subject is the effect of an intervention rather than the pattern of an association: it answers what would change if we acted, not what accompanies what in the record. Its credibility rests on the plausibility of the stand-in, which is established by research design rather than by the size of the dataset or the sophistication of the model fitted to it.

Source: Adapted from Holland (1986) and Imbens and Rubin (2015).

The closing sentence of that definition is the chapter's hinge, and it deserves to be stated at full force because it contradicts an intuition that four chapters of modeling have carefully built. Nothing in Chapters 7 through 10 improves as evidence about causes when the dataset grows. A regression on eighty thousand order lines estimates its coefficients more precisely than a regression on eight hundred, and precision is not the currency causation trades in:

an estimate can be pinned to three decimal places and still be an estimate of the wrong quantity. The missing counterfactual is not estimated more accurately by more rows of the same kind, because more rows of the same kind describe the same single world in more detail. What changes the answer is a change in how the comparison group came to be a comparison group — which is a decision about design, made before data exists, and the reason this chapter's most consequential material sits in Sections 11.4 and 11.5 rather than in the arithmetic of Section 11.6.

The practical move that rescues the situation is a shift of ambition, from the individual to the group. C004471's personal effect is unknowable, and StyleCraft does not need it. What the CMO's decision requires is the average effect across the customers the program would reach: the difference between the conversion rate this population would show if all of them received the Drop Alert and the rate the same population would show if none of them did. Both of those quantities are still counterfactual — no population is simultaneously fully treated and fully untreated — but the average has a property the individual effect does not. If StyleCraft can find or construct two groups of customers that are alike, on average, in every respect that matters for conversion, then one group's realized outcome can stand in for the other group's missing one, and the difference between their observed rates estimates the average effect. The whole apparatus of experimentation is a response to the phrase "alike, on average, in every respect that matters." Section 11.3 shows what happens when the groups are not alike, which is the ordinary condition of marketing data. Section 11.4 introduces the one procedure known to deliver the condition without requiring anyone to list what matters in advance.

11.3 Selection Bias: Naming the Confound Problem

Section 7.5 named the confounder and demonstrated it on StyleCraft's own calendar, showing that email volume and revenue correlate because the merchandising season drives both. It ended by identifying three responses — compare within the confounder, control for it in a model, or change the design so the confounder cannot operate — and it explicitly deferred the third to this chapter. This section takes up the deferral by giving the problem its causal name, showing exactly how much damage it does to the opening case's number, and explaining why the first two responses, which Chapter 7 taught, are insufficient here in a way that is not a matter of effort.

DEFINITION

Selection Bias

Selection bias is the difference between an observed group comparison and the causal effect it is taken to estimate, arising because the units that received the treatment differ systematically from those that did not in ways that themselves affect the outcome. The comparison then mixes two quantities that cannot be separated within it: the effect of the treatment and the effect of the differing composition of the groups. Selection bias is a property of how the groups came to exist, not of how carefully the outcome was measured, and it is unaffected by the size of the dataset. Its direction follows the selection mechanism rather than being fixed in advance.

Source: Adapted from Angrist and Pischke (2009) and Imbens and Rubin (2015).

In other words, the confounding of Section 7.5, seen from the causal side, becomes a question about who ended up in which group and why. And marketing manufactures selection bias as a matter of professional competence, which is what makes it so treacherous here. A CRM team that sends its best campaign to its most responsive customers is not making an error; it is doing its job, and doing it well. But the same competence guarantees that the treated group in the resulting record is composed of people who would have converted at a higher rate than the untreated group even if the campaign had never been built.

The direction of the resulting distortion deserves more care than marketing folklore usually gives it. When high-propensity customers are preferentially treated, selection commonly inflates the observed program difference, and that is the case StyleCraft's Drop Alert presents. Other targeting rules produce the opposite. A win-back program aimed at lapsed customers, a save offer aimed at accounts already showing warning signs, a discount aimed at the weakest markets, an onboarding sequence aimed at customers with no purchase history — every one of these treats

the customers least likely to convert, and the observational comparison then understates or even reverses the program's true effect. The direction of selection bias follows the selection mechanism, and an analyst who assumes it always points upward will misread half the programs she evaluates. What is general is not the sign but the fusion: the record cannot separate treatment from selection on its own. Gordon et al. (2019) put that point on the record at industrial scale, running large advertising experiments and then attempting to reproduce their answers using the observational methods practitioners rely on when experiments are unavailable, with access to a volume of behavioral covariates far beyond what most marketing organizations possess; the observational estimates diverged from the experimental ones substantially and unpredictably, in both directions and by large margins.

The Drop Alert's own numbers make the mechanism arithmetically visible, and the arithmetic is small enough to do by hand — which is where Lab 11.1 begins. Take the eight months of observational history and split it not into two groups but into four, by crossing the treatment with the one confounder StyleCraft happens to have measured: whether the customer was engaged, meaning app-installed and loyalty-enrolled, at the time of the send. The CRM team's targeting shows up immediately in the composition. Of the sixteen thousand customers in the comparison, eight thousand received the Drop Alert and 75 percent of them were engaged; eight thousand did not receive it and only 25 percent of them were engaged. And the outcomes, arranged this way, tell a different story than the headline did.

Table 11.1

The Drop Alert's observational record, arranged two ways

Group	Customers	Conversions	Conversion rate
Received, engaged	6,000	708	11.80%
Received, not engaged	2,000	96	4.80%
Received, all	**8,000**	**804**	**10.05%**
Not received, engaged	2,000	200	10.00%
Not received, not engaged	6,000	180	3.00%
Not received, all	**8,000**	**380**	**4.75%**

Read the bold rows and the deck's number appears: 10.05 percent against 4.75 percent, a difference of 5.30 percentage points, a relative lift of 112 percent. Read the unbolded rows and something more interesting is visible, twice. Among engaged customers, the treated converted at 11.80 percent and the untreated at 10.00 percent — a within-stratum observational difference of 1.80 points. Among customers who were not engaged, 4.80 against 3.00 — a within-stratum observational difference of 1.80 points. The two strata agree, which is the designed signature of a record in which the treatment did the same thing to both kinds of customer.

What the agreement licenses is narrower than it looks, and the distinction matters enough to state before the decomposition rather than after. Under this miniature's designed assumption that engagement is the only confounder, that it has been measured without error, that no unmeasured selection remains, and that treated and untreated customers are otherwise exchangeable within each stratum, standardizing on the treated group's engagement mix recovers an adjusted difference of 1.8 points. That adjusted difference is what the observational record can produce. It is not yet a causal effect, and the reason is not arithmetic but architectural: the four assumptions above are assumptions, they were installed by the example's designer, and nothing in the record can check them. The randomized experiment of Section 11.4 is what establishes the causal interpretation without requiring any of them.

With that qualification in place, the decomposition does its work. The remaining 3.5 points is composition, and it decomposes exactly. The treated group was 75 percent engaged and the untreated group 25 percent engaged, a difference in mix of 50 percentage points; engaged customers converted 7 points higher than unengaged ones absent any treatment (10.00 against 3.00); and half of a seven-point gap is 3.5 points of pure difference-in-kind, contributed by nothing but who was chosen. Two-thirds of the headline was the CRM team's targeting skill being read back to it as campaign performance. An equivalent way to see it, and the one worth carrying into the lab: ask what the treated group would have converted at if the campaign had never existed. Its own mix, at untreated rates, gives 0.75 × 10.00

+ 0.25 × 3.00 = 8.25 percent — and the treated group's realized 10.05 percent minus that constructed 8.25 percent is 1.80 points, the adjusted difference, while the constructed 8.25 percent minus the untreated group's 4.75 percent is 3.50 points, the composition component. The two quantities are additive, they are both real, and only one of them is even a candidate answer to the CMO's question.

It is tempting to conclude that the problem is solved: stratify by engagement, average the within-stratum differences, and report 1.8 points. Chapter 7 taught exactly that move, and here it worked perfectly. It worked, however, for a reason that will almost never hold in practice — the example was built with a single confounder, that confounder was measured, and it was the only one operating. Real targeting decisions are made on dozens of measured variables and at least as many unmeasured ones: the CRM manager's judgment about which customers "get" the format, the historical accident of which cohorts were enrolled during which promotion, the customer's own decision to install the app, which is itself a symptom of purchase intent that no column records. Controlling for a confounder requires having measured it, and having measured it requires having thought of it. That is the asymmetry that makes observational causal claims in marketing so unreliable: the analyst can enumerate what she controlled for, and can never enumerate what she failed to. No observational model can guarantee separation of treatment and selection without assumptions about the variables and mechanisms responsible for assignment. The lesson is not that statistical adjustment is useless. It is that adjustment can correct only the imbalances the analyst can name, while the design of Section 11.4 corrects all of them at once, including the ones nobody has thought of yet.

CONCEPT

The Comparison Is the Deliverable

By the end of this chapter the arithmetic will occupy four lines of code and the argument will occupy four pages, and the imbalance is the correct one. Two proportions and their difference is not hard mathematics; every spreadsheet in the building can compute the 5.3 points that started the argument. The question a marketing analyst is actually paid to answer is not what the difference between two groups is but whether that difference means what the meeting will take it to mean, and that question is settled entirely by how the two groups came to be groups.

Hence the working rule this chapter enforces from here forward, in the labs, in the deliverable, and in the exercises: before quoting any comparison as evidence about a decision, state in one sentence how each side of it was selected, and state what would have made the two sides differ in outcome even if the treatment had never existed. If the answer to the second question is "nothing, because a chance process assigned them," the comparison is evidence about a cause. If the answer is a list of plausible characteristics, the comparison is a description — sometimes a useful one — and the words used to report it must fall back accordingly, per Section 11.10's ladder.

Source: Course concept developed for this guide, informed by Angrist and Pischke (2015) and Gordon et al. (2019).

11.4 Random Assignment and Experiment Design

Section 11.3 ended on an asymmetry: adjustment can only correct the imbalances an analyst can name. This section introduces the procedure that corrects all of them without requiring any to be named, which is the single most important idea in the chapter and among the least intuitive, because its power comes from an act that looks, at first, like an abdication of judgment.

The procedure is to decide who receives the treatment by a chance mechanism — a coin, a random number generator, a seeded draw — with the decision made in advance and with no input whatsoever from anyone's knowledge of the customers. The mechanism admits two standard forms, and the difference between them is not pedantry, because it determines what the arm sizes are allowed to be. Under Bernoulli randomization each unit is assigned independently with a fixed probability, so the realized arm sizes are themselves random and need not come

out equal. Under complete randomization a fixed number of units is drawn at random from the eligible list, so the arm sizes are exactly what the design declared. StyleCraft's experiment used the second, and the registration says so: from the ten thousand eligible customers, a seeded random draw selected exactly five thousand for treatment, and the remaining five thousand formed the control arm. That choice is what makes the arm-size check of Section 11.5 meaningful, because a realized split of 5,000 and 4,780 under complete randomization is not bad luck — it is evidence that something happened after assignment. Nobody chose the engaged customers for the treatment arm. Nobody chose anybody for anything.

DEFINITION

Random Assignment, Treatment, and Control

Random assignment is the allocation of units to experimental conditions by a chance mechanism that is independent of every characteristic of the units, so that each unit's condition is statistically unrelated to who that unit is. The treatment group receives the intervention under test; the control group receives the existing experience, no intervention, or a declared alternative, and supplies the estimate of what the treatment group's outcome would have been in the absence of the treatment. Because assignment is independent of unit characteristics, the two groups are comparable in expectation on every characteristic — recorded or unrecorded, known or unsuspected.

The difference in outcomes is an unbiased estimate of the average effect of the treatment under conditions the design must also satisfy: assignment was implemented as planned, outcomes are observed consistently across arms, units are analyzed according to their original assignment, no unit's outcome is affected by another unit's assignment, and the treatment itself is well defined at a stated dose. Randomization delivers comparability; those five conditions deliver the estimate.

Source: Adapted from Fisher (1935) and Imbens and Rubin (2015).

The phrase carrying all the weight in the first paragraph of that definition is "recorded or unrecorded." Randomization does not make the two groups identical, and any student who checks will find that StyleCraft's treatment arm contains slightly more Gen Z customers, slightly fewer Backstage members, and a marginally different mean of prior ninety-day spend than its control arm. What randomization guarantees is that these differences arise only from the draw, are as likely to fall one way as the other, shrink as the arms grow, and — crucially — are unrelated to anything about the customers, including the twenty characteristics StyleCraft never measured and the five it does not know exist. This is why the design corrects for confounders nobody has thought of. The chance mechanism does not need to know what it is balancing. It balances everything, in expectation, because it consults nothing.

Two consequences follow, and both are routinely misunderstood in practice. The first is that randomization is a statement about the procedure, not about the resulting sample. A single experiment can draw an assignment in which the arms differ noticeably on some characteristic; the guarantee is that this is improbable and undirected, not impossible. This is exactly why the checks of Section 11.5 exist: not to "prove" the randomization worked, which no check can do, but to catch the two situations that actually occur in practice — a randomizer that was never random (assignment correlated with signup order, customer identifier range, or the hashing of a field that encodes geography) and an assignment that was random but was subsequently corrupted by an operational failure such as a targeting rule quietly re-filtering the treatment list. The second consequence is that randomization buys comparability and nothing else. It does not make the outcome metric well defined, and Chapter 3's four-element metric discipline still applies: conversion counted over what window, in what numerator, with what filters. It does not make the effect large, stable, or worth having. It does not make the experimental population representative of the rollout population — a point Section 11.13 will spend a full argument on. And it does not protect the result from the analytical mistakes of Section 11.8, all of which are committed after the assignment is complete. Random assignment removes exactly one problem, completely, and leaves every other problem in the room.

Designing the comparison, then, is mostly a matter of specifying what the control arm receives, and this is where inexperienced designs go wrong first. "Control" does not mean "nothing happens." It means the arm receives the alternative against which the treatment must prove itself, and that alternative should be the thing the business would otherwise do — which for StyleCraft is the standard drop announcement, not silence. A test of Drop Alert against silence would answer a question nobody asked (does messaging beat not messaging?) and would report a lift inflated by the entire value of ordinary communication. The control arm is the baseline, in the direct lineage of the discipline Section 2.7 introduced and Section 8.7 operationalized: the opponent the treatment must beat, chosen to represent the decision's real alternative rather than a strawman. When the real alternative is genuinely nothing — an incremental advertising test where the decision is whether to spend at all — then silence is the right control, and the arm is usually called a holdout, with the ethical questions Section 11.15 takes up.

The second design decision is the one most often made by default and most expensive to get wrong: what, exactly, is being randomized.

DEFINITION

Unit of Randomization

The unit of randomization is the entity to which the chance mechanism assigns a condition — a customer, a session, a device, a store, a geographic market, or a time period. It determines what the experiment's arms are made of and what the sample size actually is for purposes of power. Outcomes may be recorded at a finer grain than the assignment, but treatment variation exists only at the assignment grain, so effect estimates and their standard errors must respect the randomization unit or its cluster structure. The unit should be at least as large as the unit through which the treatment can reach a person, because any pathway by which one unit's treatment influences another unit's outcome undermines the comparison.

Source: Adapted from Kohavi et al. (2020).

The second sentence of that definition closes a hole that swallows a surprising number of commercial test reports. An experiment that randomized ten thousand customers does not acquire more evidence because those customers generated forty thousand orders. The orders are more observations of the same ten thousand randomizations, and an analysis that counts them as independent shrinks the interval to a width the design never earned — the grain slip Section 11.11 names among the assistant's standing failures. More order-level rows are welcome as description. They are not more randomized evidence.

The rule of thumb hidden in the definition's last sentence is that the unit must be large enough to contain the treatment's spillover. Table 11.2 lays out the marketing cases and what each choice buys and risks; it is a design checklist, and the labs and exercises return to it whenever an experiment must be specified rather than analyzed.

Table 11.2

Choosing the unit of randomization

Unit	Appropriate when	What it buys	What it risks
Session or visit	The treatment is a within-visit experience with no memory, such as a page layout tested on anonymous traffic	Very large sample sizes; fast results	The same person sees both conditions across visits, blurring the contrast and making a per-person outcome uninterpretable
Customer or account	The treatment is delivered to a person and the outcome accrues to that person, as with email, SMS, app push, or offers	A clean per-person outcome; the natural grain for CRM decisions	Spillover between customers who talk to each other; households sharing an account

Unit	Appropriate when	What it buys	What it risks
Store	The treatment is a physical or operational change, such as assortment, staffing, or in-store signage	Realism; the treatment is applied as it would really be applied	Very small sample sizes — eighteen stores is eighteen units, not eighteen thousand customers — and correspondingly weak power
Geographic market	The treatment is broadcast and cannot be aimed at individuals, as with out-of-home, local TV, or regional media	The only feasible unit for untargetable media; captures total market effect	Few units, high variance between markets, long durations, and the need for designs like those in Section 11.9
Time period	Only one unit exists and the treatment can be switched on and off, as with a single store or a site-wide policy	Feasibility where nothing else is available	Trend and seasonality confound the on and off periods; weakest of the designs listed

StyleCraft's case resolves cleanly. The Drop Alert is delivered to a person, the conversion outcome accrues to that person, and the decision under debate is a per-customer program, so the unit is the customer — which is why the ab_test file has one row per customer, and why the analysis of Section 11.6 counts customers rather than messages, sessions, or orders. Note the discipline this imposes on the metric: a customer who converts three times in the window is one conversion in a rate whose denominator is customers, per the definitional care of Section 3.5, and an analyst who quietly switches to counting orders has changed the question without saying so.

Contamination — the failure mode the unit choice exists to prevent — is treated in Section 11.8 with the other ways tests go wrong, because in practice it arrives late, as a discovery rather than a design flaw. But the design-time version of the question belongs here, and it is worth asking of every experiment before it launches: is there any path by which a treated unit's treatment can reach an untreated unit? For the Drop Alert the honest answer is that a small path exists — StyleCraft's most engaged customers follow the brand's social accounts and each other, and an early-access window that produces a visible sell-out of a style affects what remains available to the control arm hours later. The path is real, it is small at a five-thousand-customer scale against a national inventory, and the professional move is to record it in the registration as a known limitation with its expected direction (it would, if anything, depress the control arm's conversion and thereby inflate the measured lift) rather than to pretend it does not exist.

11.5 Preregistration, Power, and the Integrity of the Assignment

Random assignment makes an unbiased comparison available. It does not make an unbiased comparison inevitable, because the assignment is only half the experiment; the other half is a long series of analytical choices — which metric, which window, which customers, when to stop, which subgroups to look at — every one of which can be made after the outcomes are visible. An analyst who makes those choices while looking at the results can produce a favorable answer from a null effect essentially at will, without ever fabricating a number and usually without noticing what she is doing (Simmons et al., 2011). This section installs the discipline that closes the door, sizes the experiment that will test it, and then grades the machinery before the machinery grades the campaign.

DEFINITION

Preregistration

Preregistration is the practice of recording an experiment's design and analysis plan — the hypothesis, the primary outcome metric, the arms and their assignment, the treatment dose, the unit of randomization, the eligible population, the sample size and its justification, the planned duration, the analysis to be run, and the decision rule that maps the result to an action — in a dated, unalterable record created before the data are observed. Its function is to distinguish predictions from explanations: results specified in advance test a hypothesis, while results selected after inspection describe the data that generated them, and the two carry very different evidential weight even when the arithmetic is identical.

Source: Adapted from Nosek et al. (2018).

In other words, a preregistration is the analytic specification of Section 2.5 written for an experiment, with two elements added that Chapter 2 did not need: a stopping rule and a treatment dose. It carries the same virtues in the same order. It forces the decision to be named before the analysis begins, so that the analysis is built for a decision instead of a decision being retrofitted to an analysis. It makes the deliverable auditable, because the record can be compared afterward against what was actually done. And it protects the analyst most of all, since an experiment whose result disappoints its sponsor is precisely the experiment whose method will be litigated line by line, and the only defense that survives that meeting is a document with a date on it.

Table 11.3 is the record StyleCraft's CRM team filed before the Drop Alert experiment launched, with a third column stating why each element must be settled in advance rather than chosen afterward. Two features of it are worth reading for specifically, because both are corrections to the way the draft of this chapter first stated them. The dose row exists because a program's cost per customer is meaningless until the number of exposures is fixed, and Section 11.7's break-even line is computed from that row rather than from a sentence in a slide. And every guardrail and secondary metric named in the table is computable from the certified ab_test file as the data dictionary defines it — the converted flag, the order_revenue column, and the pre-assignment characteristics — because a preregistered guardrail that cannot be calculated is not a minor gap in enrichment but a design defect that will be discovered at exactly the wrong moment. Unsubscribe and margin fields would make richer guardrails, and StyleCraft's registration would have declared them had the messaging and merchandising systems delivered them into the certified table on the experiment's schedule. They did not, so the registration declares what the certified table can measure and names the missing fields as an instrumentation obligation for the next test.

Table 11.3

The preregistration record for the Drop Alert experiment

Element	Declared value	Why it must be declared in advance
Decision	Whether to expand the Drop Alert to the full SMS-eligible base for the Q4 drop calendar	Names the action the evidence must support, per Section 2.3
Primary metric	Conversion: share of assigned customers with at least one order in the 14 days following assignment	One metric decides; a second promoted later is a multiple comparison (Section 11.8)
Arms	Treatment: Drop Alert early access. Control: standard drop announcement	Control is the real alternative, not silence
Treatment dose	One Drop Alert, for the single drop in the customer's assignment week	Fixes the cost per assigned customer, and therefore the break-even lift (Section 11.7)

Element	Declared value	Why it must be declared in advance
Assignment mechanism	Complete randomization: a seeded draw of exactly 5,000 of the 10,000 eligible customers into treatment	Fixes what the realized arm sizes should be, so a shortfall is a finding rather than noise
Unit of randomization	Customer	Fixes the analysis grain and the true sample size (Section 11.4)
Eligible population	SMS-opted-in customers with an account at least 30 days old	Defines whom the estimate is about, and therefore the rollout it licenses
Sample size	5,000 per arm	Justified by the power calculation below; fixed in advance so the stopping rule is meaningful
Duration	Assignment window of 14 days; outcome window of 14 days after each customer's assignment	Prevents mid-test extension in search of significance
Analysis	Intention-to-treat two-proportion comparison at the customer grain; absolute lift with a 95% interval on the difference; relative lift with a separate 95% interval on the ratio	Prevents the method from being chosen to fit the result
Decision rule	Expand if the lower bound of the interval on the difference exceeds the break-even lift of 0.43 points (Section 11.7)	Converts a statistical output into an action before anyone knows the output
Stopping rule	Analyze once, after the full outcome window closes for all assigned customers	The single most effective defense against peeking (Section 11.8)
Secondary metrics	Revenue per assigned customer; order count per assigned customer	Declared as secondary and exploratory, so they cannot become the headline
Guardrails	Revenue per assigned customer in the treatment arm must not fall below the control arm's by more than 5%	Names in advance the harm that would stop a rollout even if conversion won
Instrumentation obligation	Unsubscribe events and order margin to be delivered into the certified experimental table before the next test	A guardrail that cannot be computed is a design defect, not a missing nicety

Two entries in that table are earned by arithmetic rather than by declaration, and both belong to the same idea: an experiment that cannot see the effect it is looking for is not a cautious experiment but a wasted one.

DEFINITION

Statistical Power

Statistical power is the probability that an experiment will detect an effect of a specified size, at a specified significance level, given its sample size and the variability of its outcome — the chance that a real effect of that magnitude produces a statistically significant result rather than being missed. Power rises with sample size and with the size of the effect sought, and falls as the outcome grows more variable; because the required sample size grows roughly with the inverse square of the smallest effect worth detecting, halving the detectable effect roughly quadruples the sample needed. An underpowered experiment is not merely inconclusive: its significant results, when they occur, systematically overstate the effect's size.

Source: Adapted from Cohen (1988) and Kohavi et al. (2020).

The calculation that produced StyleCraft's five thousand per arm runs in one direction only, and it starts from the business rather than the statistics. The first input is the baseline conversion rate the control arm is expected to show, which StyleCraft reads off its own history at roughly 6 percent. The second is the minimum detectable effect: the smallest lift that would change the decision, which is a business judgment, not a statistical one, and which the CRM team set at 1.5 percentage points on the grounds that anything smaller would not justify the operational overhead of running early access across a national drop calendar. The third is the significance level, conventionally 5 percent, and the fourth is the power target, conventionally 80 percent. With those four numbers, the standard sample-size calculation for a difference in proportions returns 4,379 customers per arm; the team rounded to 5,000 for operational tidiness and to buy margin against the delivery failures that always occur. Code 11.7 makes the calculation executable rather than asserted, which is where students should verify every figure in this paragraph.

Read the calculation backward and it tells you something more useful than a number. At 5,000 per arm, StyleCraft has a 94.5 percent chance of detecting the 1.8-point effect the generator planted; at 1,000 per arm it would have had a 35.6 percent chance, meaning a two-in-three probability of running a perfectly designed experiment on a real effect and concluding there was nothing there. The inverse-square relationship stated in the definition is the fact worth memorizing, because it is what makes small effects expensive: detecting half a point instead of 1.5 requires not three times the sample but roughly eight and a half times it, near 36,800 customers per arm, which for many marketing questions exceeds the entire addressable base. Lewis and Rao (2015) turned this arithmetic into a general result about advertising, showing that because sales outcomes are so variable relative to advertising's true effects, the experiments needed to measure advertising returns with usable precision are far larger than the ones firms typically run — which means many published advertising ROI figures are drawn from tests that never had the power to produce them.

With the size settled, three instruments grade the assignment rather than the campaign, and the order in which this chapter presents them is the order the deliverable follows. The first is the intention-to-treat rule, which decides who is in the comparison at all.

DEFINITION

Intention-to-Treat Analysis

Intention-to-treat analysis compares units according to the condition they were randomly assigned, regardless of whether the assigned treatment was actually delivered, received, or complied with. Its purpose is to preserve the property random assignment created: because assignment was independent of every unit characteristic, groups defined by assignment remain comparable, while groups defined by what happened after assignment — messages that reached a handset, customers who opened an app — are no longer randomized and reintroduce exactly the selection the design removed. The intention-to-treat estimate answers what the effect of offering the program is; the effect of the treatment actually received is a different quantity requiring stronger assumptions.

Source: Adapted from current trial-reporting guidance on retaining randomized participants in their assigned groups for the primary analysis (Hopewell et al., 2025) and Imbens and Rubin (2015).

This corrects a reflex the draft of this chapter encouraged and every experienced trialist would reject. When a 5,000/5,000 design reports 5,000 treatment customers and 4,780 control customers, the temptation is to conclude that 220 customers were "lost" and that the comparison is therefore invalid. The right reading is more precise and less alarming. A delivery failure is important operational evidence, and it is not by itself grounds for removing anyone from the primary comparison; removing the customers whose messages failed is precisely how an analyst converts a randomized comparison back into an observational one, because message delivery correlates with handset, carrier, tenure, and engagement. What the shortfall does demand is an explanation, and the instrument that produces it is a flow record.

Table 11.4

The assignment-flow record every experiment report must carry

Stage	Treatment	Control	What a gap at this stage means
Randomized	5,000	5,000	The design's declared allocation under complete randomization; any shortfall here is an assignment-code or list-construction defect
Message successfully delivered	reported from the messaging log	not applicable; control receives the standard announcement	Operational evidence about reach. It is reported, investigated, and never used to filter the primary comparison
Primary outcome observed	5,000	5,000	Outcome ascertainment. A gap here is the serious one: it means conversion could not be determined for some assigned customers
Included in the intention-to-treat analysis	5,000	5,000	Everyone randomized with an observed outcome, in the arm they were assigned to. This is the row the lift is computed from

Read the table as three different estimands wearing similar clothes. The intention-to-treat effect is the effect of assignment to the Drop Alert program, which is what a rollout decision actually needs, because a rollout also delivers imperfectly. The effect of treatment received is a different quantity, estimable only under assumptions about why delivery failed, and it is out of scope for this guide. And a missing outcome — a customer randomized whose conversion cannot be determined at all — is neither of those; it is a defect requiring investigation and, where it cannot be repaired, a sensitivity analysis that shows how the conclusion moves under the best and worst plausible assumptions about the missing customers. Only the last of these three licenses anyone to remove a randomized customer from a table, and even then it is reported as a deviation.

The second instrument compares pre-assignment characteristics across arms, and it is the one most often misused. The draft of this chapter implied that a prior-spend difference of more than a percent or two between arms would invalidate the simple randomized comparison. It would not, and reasoning that way inverts the logic of the design. Random assignment does not guarantee balance in every realized sample; chance imbalance can occur when the procedure is entirely valid, and baseline significance tests or arbitrary imbalance thresholds should not be used to decide retrospectively whether a valid randomization "worked" (Altman, 1985; Senn, 1994). A conspicuous imbalance is a diagnostic signal, not automatic invalidation. It should trigger an audit of the assignment code and its log, a check that no post-assignment filtering occurred, and a look at whether the imbalance appears across many variables or only one — a single covariate out of line with all the others, especially one related to geography or signup date, is the fingerprint of a randomizer keyed to something structured rather than random. If the randomization record is intact, chance imbalance does not remove the causal basis of the intention-to-treat comparison, and the correct optional response is a covariate-adjusted analysis declared in advance for precision, not a re-randomization, which would itself introduce selection.

The third instrument is the one that grades the pipeline rather than the sample.

DEFINITION

A/A Test

An A/A test is an experiment in which both arms receive the identical experience, run through the full assignment, delivery, and analysis pipeline. Because there is no treatment, the true effect is zero, and the test therefore grades the apparatus rather than the intervention: assignment rates should match the intended split, pre-assignment characteristics should balance, and the measured difference should be statistically significant at approximately the nominal rate rather than more often. Because the nominal rate is not zero, a single significant A/A result is an occasion for investigation rather than a verdict; repeated or systematic failures are strong evidence of a defect in randomization, delivery, or measurement.

Source: Adapted from Kohavi et al. (2013).

The distinction the definition insists on matters because this chapter's lab does something narrower than a genuine A/A test, and the draft conflated the two. A true A/A test sends both arms through the production randomizer, the production message delivery, the production tracking, and the production campaign configuration, and it must be run before or alongside real experiments to certify that machinery. What Lab 11.1 Part B does is take completed control-arm outcomes and divide them at random into two sham groups. That procedure checks the analysis path — the metric definition, the test statistic, the code — and it checks the false-positive behavior of the test under a known-zero effect. It checks nothing whatever about the randomizer, the delivery, the tracking, or the assignment-to-exposure link, because none of those touched the sham split. This guide therefore calls the lab procedure a sham-split analysis-path check, reserves "A/A test" for the end-to-end version, and asks students to state which of the two they have run whenever they report one.

CONCEPT

Declare It or Discover It, Never Both

Every element of an experiment is either declared before the outcomes are visible or discovered after them, and the same number means different things depending on which. A 1.8-point lift on the metric named in Table 11.3 is a tested prediction. A 1.8-point lift on the metric selected from among nine after the results arrived is a description of the largest of nine numbers, and its interval — computed as if it were the only one — is wrong by an amount nobody can calculate.

The rule this guide enforces is not that discovery is forbidden; exploration after an experiment is legitimate, valuable, and how the next hypothesis gets built. The rule is that discovered findings are labeled as discovered, reported in the exploratory section of the deliverable, never given a p-value that pretends they were predicted, and never used as the basis for a decision until a subsequent experiment declares them in advance and tests them properly. Two words on a slide — "primary" and "exploratory" — carry the entire distinction, and an analyst who applies them honestly will occasionally lose an argument in the short run and will never have to retract a rollout.

Source: Course concept developed for this guide, informed by Nosek et al. (2018) and Simmons et al. (2011).

11.6 Measuring Lift

The design is now complete and the outcomes are in: the experiment ran, the outcome window closed, and the ab_test file contains one row per assigned customer with the arm, the pre-assignment characteristics, and the conversion outcome. This section computes the answer. It is the shortest conceptual section in the chapter, because

the analysis of a well-designed experiment is deliberately simple — the credibility was purchased in Sections 11.4 and 11.5, and the arithmetic merely collects it.

Both arms are counted at the grain of assignment, and every randomized customer with an observed outcome is counted in the arm she was assigned to. Of the 5,000 customers assigned to control, 300 converted within the fourteen-day window, a rate of 6.00 percent. Of the 5,000 assigned to treatment, 390 converted, a rate of 7.80 percent. Everything the chapter needs follows from those four numbers, and the first decision is how to express the difference between them, because there are two conventions and they produce numbers that differ by a factor of seventeen.

DEFINITION

Absolute Lift and Relative Lift

Absolute lift is the difference between the treatment and control rates, expressed in the units of the metric — for a conversion rate, in percentage points. Relative lift is that same difference expressed as a proportion of the control rate, usually reported as a percentage change. Absolute lift is the quantity that scales to business volume, since multiplying it by the number of customers reached yields the incremental count directly. Relative lift expresses the effect against its own baseline, which can support comparison across settings when a multiplicative-effect assumption is plausible; transport across populations, periods, or doses remains a substantive assumption rather than a property of the statistic. Relative lift is systematically the larger and more persuasive of the two whenever the baseline rate is small, and reporting either without the other, or without the baseline rate, invites misreading.

Source: Adapted from Kohavi et al. (2020).

For the Drop Alert, the absolute lift is 7.80 minus 6.00, or 1.80 percentage points, and the relative lift is 1.80 divided by 6.00, or 30 percent. Both are correct. The deck that opens this chapter reported a relative number for the observational comparison — "more than 110 percent" — and the choice was not innocent, because a relative lift on a small base is the largest, most quotable version of any marketing result and the one least connected to money. The guide's standing rule from here forward: report the absolute lift first, in percentage points, with the control rate stated beside it, then the relative lift, then the translation into incremental conversions and dollars at the rollout's scale. An analyst who reports "30 percent lift" and stops has told a stakeholder almost nothing, since 30 percent of an unstated base is not a quantity.

The definition's middle sentence corrects a claim the draft of this chapter made too strongly. It is often said that relative lift is the quantity that transfers across populations with different baseline rates, and the sentence has the form of a statistical fact while actually being an empirical hope. A relative lift transfers only if the treatment's effect really is multiplicative in the baseline — if a program that raises a 6 percent rate by 30 percent would also raise a 2 percent rate by 30 percent. Sometimes that holds approximately. Sometimes the effect is closer to additive, and sometimes it is neither, because the mechanism that makes a treatment work in one population is absent in another. The honest position is that relative lift can support cross-setting comparison under a stated multiplicative assumption, and that the assumption belongs in the deliverable's limitations rather than in the definition of the statistic.

The estimate now needs its uncertainty, and this is where Section 7.10's evidence reading extends to a new kind of statistic rather than being re-derived. The logic is identical: 1.80 points is an estimate from one draw of ten thousand customers, another draw would have produced a somewhat different number, and the standard error measures that wobble. What differs is only the formula, because the outcome is a proportion rather than a continuous spend figure: the standard error of a difference in two proportions is built from each arm's rate and size, and for StyleCraft's numbers it works out to 0.5066 percentage points. The 95 percent confidence interval on the difference is the estimate plus or minus roughly two standard errors, which gives a range from 0.81 to 2.79 percentage points. The corresponding test of the null hypothesis that the two population rates are equal returns a test

statistic of 3.55 and a two-sided p-value of 0.0004 — the data are hard to reconcile with "the Drop Alert does nothing" — and the guide's position on that p-value is exactly the position Section 7.10 took, adopted from the American Statistical Association's consensus statement: it is a measure of surprise under the null, not the probability that the finding is true, not a measure of the effect's size, and not by itself a warrant for a decision (Wasserstein & Lazar, 2016).

What carries the decision is the interval, and reading it well is the section's real skill. The interval says that lifts as small as 0.81 points and as large as 2.79 points are compatible with what the experiment observed. That is a range of more than three to one in business terms, which is worth sitting with: a well-powered experiment on ten thousand customers, correctly designed and cleanly executed, still leaves the effect known only to within a factor of three. Two habits follow. The first is to check where the true value sits — and here the chapter's verification theme pays off, because the designed lift of 1.8 points that the generator planted lies comfortably inside the interval, near its center, which is what a correct analysis of a correct experiment should produce and what the lab requires students to confirm. The second is to translate both ends of the interval into the decision's units before drawing any conclusion, because a range that is wide in percentage points may be irrelevant in dollars or decisive in dollars depending entirely on the program's economics. That translation is Section 11.7.

Two technical notes belong here and then get out of the way, and the second is a correction the draft needed. First, the interval above uses each arm's own rate to estimate the standard error, while the significance test conventionally pools the two arms under the null; for StyleCraft's numbers the two give 0.5066 and 0.5069 percentage points respectively, a difference visible only in the fourth decimal, and the guide does not ask students to choose between them beyond knowing that the software's default is one of the two and that both are reported in the lab. Second, an interval for the relative lift may not be obtained by dividing the absolute interval's endpoints by the observed control rate. That shortcut treats the control rate as a known constant when it is itself an estimate with its own sampling error, and it therefore produces an interval that is too narrow and not centered where a ratio interval should be. For StyleCraft's numbers the shortcut returns roughly 13 percent to 47 percent, while a properly constructed 95 percent interval for the ratio of the two proportions — computed on the log scale and exponentiated, which is the standard construction — runs from a risk ratio of 1.12 to 1.50, or a relative lift of roughly 12 percent to 50 percent. The difference is not large here, and the discipline is the point: a confidence interval for a ratio is computed as a ratio. Code 11.9 computes both and prints them side by side so students can see how far the shortcut lands from the honest answer. The absolute lift remains the primary result throughout, and the relative interval is reported second, precisely because a stakeholder who has been told "30 percent" will be surprised to learn the evidence is consistent with 12.

11.7 Statistical Significance, Practical Significance, and the Break-Even Line

Section 11.6 produced an interval and a p-value. This section converts them into a decision, and the conversion turns on a distinction the guide has raised before in weaker forms and now makes operational: the difference between a result the data can distinguish from zero and a result the business should act on. The two questions are unrelated, they are answered by different quantities, and confusing them is the most common analytical error in commercial experimentation.

Statistical significance answers a question about evidence: given the sample, is the observed difference larger than the wobble that random assignment alone would produce? Practical significance answers a question about consequences: is a difference of this size worth what it costs to obtain? The first is settled by the interval's distance from zero; the second is settled by the interval's position relative to a line the analyst must compute from the program's economics before the experiment reports anything. Both failure directions occur regularly. A very large test — hundreds of thousands of customers, as platform experiments routinely run — will return significant results for effects of one-hundredth of a percentage point, differences that are unambiguously real and completely worthless. A small test on a genuinely valuable treatment will return a wide interval straddling zero, and the correct reading is not "the treatment does nothing" but "this experiment could not tell," which is a statement about the experiment, per Section 11.5's power discussion.

The instrument that makes practical significance concrete is the break-even lift, and it cannot be computed at all until the treatment's dose is fixed.

CONCEPT

The Break-Even Lift, and Why the Dose Comes First

Every marketing program has a cost per customer reached and a contribution per incremental conversion, and their ratio is the lift at which the program exactly pays for itself. The ratio is undefined until someone states how many times the program touches a customer, because the cost per customer reached is a cost per exposure multiplied by a number of exposures. The Drop Alert costs twelve cents per alert; Table 11.3 declares a dose of one alert per treated customer, delivered for the single drop in the customer's assignment week, with conversion measured over the fourteen days that follow. Total cost per assigned customer is therefore $0.12, StyleCraft's average order contributes about $28 after product cost, and the break-even lift is $0.12 ÷ $28 = 0.0043, or 0.43 percentage points.

Change the dose and the line moves. Had the test delivered two weekly alerts across its fourteen-day window, the cost per assigned customer would have been $0.24, the break-even lift would have been 0.86 points, and the same measured interval would no longer clear it at its lower end — the recommendation would change, on identical evidence, because the program being priced would be a different program. This is why the dose belongs in the registration beside the metric, and why the rollout's dose must be stated and compared: an experiment run at one alert per drop licenses a rollout at one alert per drop, and a rollout that raises the frequency is a different intervention whose economics and whose response must both be re-derived.

The comparison that follows is made against the whole interval, not the point estimate. If the interval's lower bound exceeds the break-even line, the program pays even in the pessimistic world the data still permits, and the decision is robust to the uncertainty rather than dependent on the estimate being exactly right. If the line falls inside the interval, the honest report is that the evidence does not settle the question, and the choice belongs to the decision-maker under the asymmetric-cost reasoning of Section 2.7 — which is a different sentence, and a more defensible one, than a recommendation dressed in a p-value.

Source: Course concept developed for this guide, informed by Kohavi et al. (2020).

Run the comparison for the Drop Alert and the decision becomes unusually clean. The break-even lift is 0.43 points. The experiment's interval runs from 0.81 to 2.79 points. The entire interval sits above the line, which means that every value of the effect the data considers plausible is a value at which the program pays for itself — and the recommendation therefore does not depend on the point estimate being right. Priced per thousand customers enrolled at the tested dose, the arithmetic reads: at the point estimate of 1.80 points, eighteen incremental conversions per thousand, contributing $504 against a program cost of $120, a net of $384; at the interval's pessimistic edge of 0.81 points, roughly eight incremental conversions, contributing $226 against the same $120, a net of $106; at the optimistic edge of 2.79 points, about twenty-eight conversions, $782 against $120, a net of $662.

That arithmetic is exact and its inputs are not, which the deliverable must say in the same breath. Below the break-even lift the program loses money and above it the program makes money under a specific set of assumptions, and the phrase "at any scale" that the draft of this chapter attached to those clauses is not one the arithmetic supports. The calculation assumes that treatment cost is linear in customers reached, that contribution per incremental conversion is stable, that there is no fixed implementation cost to amortize, that no capacity or inventory constraint binds at rollout volume, that order value and margin do not change, that unsubscribe and fatigue costs are zero, and that response does not diminish as the program scales. Every one of those assumptions is plausible at the tested dose

and volume, and several of them are known to weaken at rollout. The $28 contribution figure deserves particular scrutiny, because it is an average across observed orders, and the orders an incremental converter places need not carry the same category mix or margin as the average order. That is exactly why revenue per assigned customer is a declared secondary outcome in Table 11.3 rather than an afterthought, and why the deliverable presents the break-even comparison with a sensitivity band — the line recomputed across a plausible range of contribution and cost — rather than as a single number. Code 11.12 prints that band.

The honest sentence for the CMO is therefore not "the Drop Alert lifts conversion 30 percent" but something closer to: at a dose of one alert per drop, the program is profitable across the full range of effects the experiment supports, with a most likely value near $384 per thousand customers per drop window and a plausible range from roughly $106 to $662 under the stated cost and contribution assumptions — and the decision to expand does not hinge on which end proves true.

That is a strong result, and this is the moment to notice what it did not license. Three restrictions travel with it, all of which Section 11.13 will develop into the deliverable's language and none of which the p-value or the interval expresses.

The first is population. The experiment ran among SMS-opted-in customers with accounts at least thirty days old, which is a population selected for exactly the engagement that predicts response. The CMO's proposed rollout reaches the full SMS-eligible base, which is broader. Nothing in the experiment estimates the effect among customers outside its eligible population, and the reasonable prior — supported by the observational table of Section 11.3, where the adjusted difference happened to be identical in both engagement strata but need not have been — is that the effect on less engaged customers could be smaller, larger, or differently signed. The estimate is about the tested population, full stop, and the extrapolation to a broader one is an assumption the deliverable must name rather than a finding the experiment supports.

The second is dose and duration, and Section 11.7's break-even discussion has already shown how much of the business case rides on it. The experiment measured one alert and one fourteen-day outcome window in a non-holiday period. The rollout would run for a quarter, through the holiday calendar, at an alert frequency the CRM team has proposed increasing. Effects measured at one exposure routinely fail to survive a frequency increase, because unsubscribe and fatigue are nonlinear, and effects measured over two weeks routinely fail to persist over three months, for reasons Section 11.8's novelty discussion makes precise. Doubling the dose doubles the cost with certainty and does not double the response with anything like the same certainty, which is a specific and testable claim the rollout plan must schedule a measurement for.

The third is the counterfactual on the other side of the budget. The CMO's proposal moves four hundred thousand dollars out of paid social. The experiment tells StyleCraft what the Drop Alert is worth; it says nothing whatever about what the paid social spend is worth, and comparing a measured program against an unmeasured one systematically favors the measured one — which, as Section 11.14 will show, is one of the most consequential biases in modern marketing budgeting. The correct deliverable states plainly that the reallocation decision requires an incrementality estimate for paid social that StyleCraft does not currently have, and recommends the experiment that would produce it. This is not evasion. It is the analyst refusing to let a well-measured 1.8 points beat an unmeasured number by default.

11.8 How Tests Go Wrong

An experiment's credibility is established by design and destroyed by execution, and the destruction is almost always inadvertent. This section catalogs the five failures that account for most of the wreckage in commercial testing, each with the mechanism that produces it, the symptom it leaves in a report, and the defense. The first two are analytical, committed by the person reading the results; the third is operational, committed by the team running the test; the fourth and fifth are substantive, committed by reality.

The first is peeking, and it is the most common and the most misunderstood.

DEFINITION

Peeking

Peeking is the practice of examining an experiment's results before the pre-declared sample size or duration has been reached and allowing what is seen to influence when the experiment stops or how it is analyzed. Because the estimate fluctuates as data accumulate and each look offers a fresh opportunity to observe a fluctuation that crosses the significance threshold, the probability of declaring a difference where none exists rises with the number of looks — far above the nominal level the test's p-value reports. The resulting false positives are not detectable in the final output, since the test statistic computed at the moment of stopping looks entirely ordinary.

Source: Adapted from Johari et al. (2017).

The mechanism deserves to be felt rather than merely stated, which is why Lab 11.2 makes students manufacture a false winner from StyleCraft's own control arm. Imagine a test in which the treatment does nothing at all. On any single look at the accumulated data, there is a 5 percent chance the difference crosses the conventional threshold by luck. An analyst who looks once and stops has a 5 percent false-positive rate, as advertised. An analyst who checks the dashboard each morning for two weeks and stops the moment the result "reaches significance" has taken fourteen opportunities to catch a 5 percent event, and while the looks are correlated — each day's data contains the previous day's, so the arithmetic of fourteen independent trials, which would suggest 51 percent, badly overstates the damage — the inflation remains severe. On StyleCraft's own control arm at fourteen evenly spaced looks, the probability of seeing a significant difference in either direction at some point rises to roughly 22 percent, more than four times the nominal rate, and Lab 11.2 has students measure it.

The organizational version is worse than the statistical one, because the stopping is asymmetric. Nobody stops a test early because the treatment is losing; tests are stopped early because the treatment is winning, which converts peeking from a symmetric random error into a machine that manufactures winners. That asymmetry is worth measuring separately rather than folding into the two-sided figure, and the lab does so: the probability of at some point seeing a significant difference with the treatment arm ahead — the policy an actual analyst follows — runs near 11 percent on this design, roughly one test in nine. The two-sided 22 percent and the favorable-only 11 percent answer different questions, and reporting one while describing the other is a small dishonesty the draft of this chapter committed and Code 11.10 now separates.

The defense declared in Table 11.3 is the simplest available: fix the sample size and the duration in advance, look once when the outcome window closes, and treat the dashboard between launch and close as an operational monitor for delivery failures rather than a scoreboard. Sequential methods that permit valid continuous monitoring exist and are standard at platform scale (Johari et al., 2017), but they must be declared in advance too — a test analyzed with an ordinary two-proportion comparison after ad hoc looking is not rescued by mentioning that such methods exist.

The second failure is peeking's twin, committed across metrics rather than across time.

DEFINITION

Multiple Comparisons

The multiple comparisons problem is the inflation of false-positive risk that occurs when many hypotheses are tested and any significant result among them is treated as though it had been the only test performed. With independent tests at a 5 percent threshold, the chance of at least one spurious significant result is roughly 64 percent after twenty tests and rises toward certainty thereafter, so a "winning" metric or segment selected from a large set carries far weaker evidence than its p-value suggests. Remedies include declaring a single primary outcome in advance, adjusting thresholds for the number of comparisons, and treating unplanned findings as hypotheses for a future experiment rather than as results.

Source: Adapted from Simmons et al. (2011) and Kohavi et al. (2020).

In marketing the multiplicity is rarely deliberate and almost always large. A single experiment report may carry conversion, revenue per customer, average order value, order count, app opens, and margin — six metrics — each cut by four segments, two channels, and three loyalty tiers, which is not six tests but well over a hundred. Somewhere in that grid, something is significant. The symptom in a report is unmistakable once you know to look for it: a headline that describes a subgroup rather than the population ("the Drop Alert lifted conversion 6.2 points among Gen Z suburban app users"), attached to an interval computed as if that subgroup had been the plan all along. The defense is Table 11.3's primary-metric row and Section 11.5's labeling discipline. One primary metric decides. Everything else is exploratory, is labeled exploratory in the deliverable, and generates the next preregistration rather than this quarter's rollout.

The third failure is operational, and it is the reason experiment platforms have change logs. Mid-test changes are alterations to the treatment, the control, the eligible population, or the measurement while the experiment is running: a creative refresh applied to the treatment arm in week two, a bug fix that changes what the control experiences, an eligibility filter tightened after launch, a tracking change that alters how conversions are counted. Each of them silently redefines what is being compared, so that the final result estimates the effect of neither the original treatment nor the revised one but of an undocumented blend whose composition depends on when customers happened to be assigned. The symptom is a result that changes character across the test window without an obvious substantive reason, and the defense is a change log plus a hard rule that any change to the treatment restarts the experiment. StyleCraft's registration handles the common marketing version of this failure explicitly: the eligible population and the message creative were fixed before launch, and the CRM team's ordinary weekly optimization was suspended for both arms during the test window.

The fourth failure is substantive, and it is the one most likely to cost a marketing organization real money after a correct experiment.

DEFINITION

Novelty Effect

A novelty effect is a response to a treatment that arises from its unfamiliarity rather than from its enduring value, producing an initial effect that decays as the audience habituates. Its mirror image, sometimes called a primacy effect, is an initial underperformance caused by disruption of established habits, which fades as users adapt. Because both concern how response changes with repeated exposure or with time since first exposure, detecting either requires outcomes observed across multiple exposures or multiple periods for the same randomized units — not a comparison of units first exposed at different dates.

Source: Adapted from Kohavi et al. (2020).

The Drop Alert is an unusually good candidate for the effect. A twelve-hour early-access window is interesting the first time a customer receives it and steadily less interesting the twentieth time, particularly if the customer discovers that the styles she wanted were still available at the general announcement anyway. An experiment measuring one alert and one fourteen-day window is therefore measuring the alert at its most novel, and the honest expectation is that a quarter-long rollout at higher frequency returns something smaller than 1.8 points.

What the closing sentence of the definition rules out is the diagnostic the draft of this chapter proposed, and the correction is worth stating plainly because the flawed version is everywhere in commercial practice. Splitting an experiment's customers into those assigned in the first half of the assignment calendar and those assigned in the second, then comparing the estimated lift in each, does not test for novelty. It compares two different cohorts of customers, assigned on different dates, exposed to different drops, under different promotional and calendar conditions. Any difference between the halves can be cohort mix, drop attractiveness, promotion timing, weather, or — given roughly 2,500 customers per arm per half — plain noise. What the comparison can honestly be called is a

temporal-stability diagnostic: a crude check for whether the estimate wobbles alarmingly across the window, useful only for flagging a difference dramatic enough to demand investigation and incapable of certifying that no decay exists. This guide labels it that way, and Code 11.13 prints it under that name.

Estimating a novelty effect properly requires a design the certified ab_test file does not support, and saying so is more useful than pretending otherwise. The file carries one row per customer with a single binary outcome over a single fourteen-day window, which contains no repeated-exposure information at all. The design that would answer the question retains one randomized assignment across several consecutive drops, records the outcome after the first exposure, the second, the third, and so on, estimates a treatment-by-period interaction, and schedules a longer-run follow-up after the program has stopped being new. That design is what the rollout plan must commission, and Exercise 11.4 asks students to specify it.

The fifth failure is the design-time contamination question of Section 11.4, arriving as an execution problem.

DEFINITION

Contamination

Contamination — also called interference or spillover — occurs when a unit's outcome is influenced by the treatment assigned to other units, violating the assumption that each unit's outcome depends only on its own assignment. Common marketing pathways include treated customers sharing offers with untreated ones, treated and untreated customers competing for the same limited inventory, household or device sharing that mixes arms, and market-level effects such as auction prices that respond to a treated group's behavior. Contamination biases the measured difference — usually toward understating the treatment's true effect when the control arm partially receives it, and toward overstating it when the treated arm's activity depletes something the control arm needed — and it is addressed by choosing a larger unit of randomization rather than by any adjustment made after the fact.

Source: Adapted from Kohavi et al. (2020).

Marketing's canonical contamination case is the promotion code that leaks: a treatment-arm offer posted to a deals forum and redeemed by hundreds of control-arm customers, which pulls the two arms toward each other and shrinks the measured effect toward zero. The symptom is a suspiciously small effect combined with control-arm behavior that should not have been possible, and the defense at design time is either a larger unit — household, store, market — or a treatment that cannot travel, such as a personalized link. StyleCraft's version, noted in Section 11.4, runs the other way: early access to limited inventory could deplete what the control arm finds later. Its expected direction was recorded in the registration precisely so the estimate could be read with the bias's sign known rather than debated afterward.

Table 11.5 collects the five failures with their mechanisms, their symptoms, and their defenses. It is written to be used rather than read: the symptom column is what an analyst actually encounters — somebody else's test report, a vendor's case study, a slide from another team — and Exercise 11.7 is nothing but this column, applied to six sentences.

Table 11.5

The five standing ways marketing tests go wrong

Failure	Mechanism	Symptom in the report	Defense
Peeking	Repeated looks at accumulating data, with asymmetric early stopping on favorable results	A test that stopped at an unplanned moment; a duration that does not match the plan; a winner that does not reproduce	Fixed sample size and duration; a single analysis at close; declared sequential methods if monitoring is required

Failure	Mechanism	Symptom in the report	Defense
Multiple comparisons	Many metrics and segments tested, the largest reported as if planned	A headline about a subgroup; a p-value attached to a finding nobody predicted	One declared primary metric; everything else labeled exploratory and re-tested before it is acted on
Mid-test changes	Treatment, control, population, or measurement altered while the test runs	Effects that shift character mid-window; a definition that changed after launch	Change log; freeze on optimization during the test; any treatment change restarts the experiment
Novelty effect	Response driven by unfamiliarity, decaying with repeated exposure	A strong single-exposure effect that fades after rollout at higher frequency	A design that follows one randomized assignment across several exposures; a treatment-by-period estimate; a planned re-measurement after habituation
Contamination	Treatment reaches or affects control units	An implausibly small effect; control-arm behavior the design should have prevented	Larger unit of randomization; non-transferable treatments; expected direction declared in advance

11.9 When You Cannot Randomize: Quasi-Experimental Intuition

Everything so far has assumed the analyst can decide who is treated. Frequently she cannot. A store's assortment change applies to the store; a regional out-of-home campaign applies to the metro; a pricing policy applies to the catalog; a competitor's entry applies to whoever it applies to. This section sketches, carefully and at concept level, what can be salvaged when randomization is unavailable — and it is bounded deliberately, because the designs below are the entry point to a large technical literature and this guide teaches only enough to read such evidence critically and to recognize when a specialist is required.

Begin by disposing of the weakest design, which is also the most common in marketing decks. A simple before-and-after comparison — revenue in the eight weeks after the campaign against the eight weeks before — is a causal claim with no comparison group at all, and its implicit counterfactual is that nothing else would have changed. In a business with the seasonality of Section 10.3, an expansion producing structural breaks per Section 10.4, and a promotional calendar that moves demand around inside the quarter, that counterfactual is not merely uncertain but usually false. A pre/post comparison is worth computing and worth never reporting alone.

The repair is to add a comparison group that experienced the same period without the treatment: other stores, other metros, other customer cohorts. The comparison group supplies an estimate of what would have happened anyway, and the treated group's change is judged against it rather than against zero. This is difference-in-differences, and it is best learned as arithmetic before it is learned as a method.

DEFINITION

Difference-in-Differences

Difference-in-differences estimates a treatment effect by comparing the change in outcomes over time in a treated group with the change over the same period in an untreated comparison group, and taking the difference between the two changes. The comparison group's change stands in for what the treated group's change would have been absent the treatment, so any level difference between the groups that is stable over time is removed by the design. The estimate's validity rests on the parallel trends assumption: that in the absence of the treatment, the two groups' outcomes would have moved in the same direction by the same amount. This assumption concerns an unobserved counterfactual and therefore cannot be verified — only supported, most commonly by showing that the groups' outcomes moved together over multiple periods before the treatment began.

Source: Adapted from Card and Krueger (1994) and Angrist and Pischke (2015).

Suppose StyleCraft's suburban New York metro receives a regional out-of-home campaign that cannot be aimed at individuals, and the analyst designates the Boston and Chicago flagship markets as the comparison. Revenue per active customer in the treated metro runs $82 in the eight weeks before and $95 in the eight weeks after, a change of $13. In the comparison markets it runs $78 and $86, a change of $8. The difference-in-differences estimate is $13 minus $8, or $5 per active customer — and the $8 is the whole point of the exercise, since a pre/post report would have credited the campaign with $13 and quietly attributed the season, the assortment refresh, and the national brand momentum to a billboard.

What the design assumes is where the caution lives. Parallel trends is a claim about a world that did not happen, and it fails routinely in exactly the situations marketing cares about most: the metro chosen for the campaign was often chosen because something was already happening there, which is selection bias operating on markets rather than customers, and the comparison markets differ in store maturity, competitive intensity, and customer mix in ways that may make their trends diverge for reasons unrelated to any treatment. The evidence that supports the assumption — never proves it — is the pre-period, and the standard practice is to plot several periods of both groups before the treatment and confirm they move together. If they do not, the design is not usable and no amount of statistical machinery repairs it. Athey and Imbens (2017) survey the modern methods that extend this family, and their existence should be read as a signal rather than a menu: these designs are a specialty, their assumptions are contestable, and a marketing analyst's job is usually to recognize that a quasi-experimental claim is being made, ask what the comparison group is and whether the pre-trends were shown, and know when to bring in help.

The reason the section is bounded so tightly is worth stating plainly, because the temptation runs the other way. Quasi-experimental designs are attractive precisely because they promise causal language without the operational cost of an experiment, and that promise is what makes them dangerous in a marketing organization. Their conclusions rest on assumptions that cannot be checked, held by analysts who are motivated to believe them, about programs their own colleagues designed. The guide's working position: prefer an experiment whenever one is possible, use these designs when it is not, state the assumption in the deliverable in the same font as the result, and treat the resulting evidence as occupying the rung below a well-executed experiment on the ladder that Section 11.10 now builds.

11.10 The Evidence Hierarchy

This chapter began by distinguishing prediction from causation, and it ends its conceptual arc by making the distinction usable. The evidence hierarchy is a ranking of the kinds of support a marketing claim can have, and its purpose is not to rate methods but to constrain vocabulary: each rung licenses a specific set of verbs, and using a higher rung's verbs on a lower rung's evidence is the single most common form of analytical overclaiming, as Section 7.15 argued at length.

DEFINITION

Evidence Hierarchy

The evidence hierarchy is an ordering of sources of evidence by the strength of the causal claim each can support: anecdote and observation at the base, then measured association, then association estimated with statistical controls or predictive validation, then quasi-experimental designs resting on stated assumptions, and finally randomized experiments at the top. Each rung narrows the set of rival explanations that remain live. The hierarchy describes the warrant available from competently executed designs; design labels alone do not determine evidential quality. It is a guide to what may be asserted, not to what is useful — evidence at every rung has legitimate roles, including generating the hypotheses that higher rungs test — and its practical function is to hold the language of a conclusion to the strength of the design that produced it.

Source: Adapted from Angrist and Pischke (2015) and Kohavi et al. (2020).

That qualification in the middle of the definition is not diplomatic hedging; it is the difference between a hierarchy and a badge. A randomized experiment with severe outcome attrition, widespread noncompliance, uncontrolled interference between arms, a primary metric switched after the results arrived, or a stopping decision made on the fourteenth peek can easily provide weaker evidence than a carefully executed difference-in-differences whose pre-trends were shown and whose assumption was stated. The rung a study occupies is set by its design; the warrant it actually delivers is set by its design and its execution together. This is why Table 11.4's assignment-flow record and Section 11.5's integrity checks are not paperwork attached to the top rung — they are what entitles a study to claim it.

Table 11.6 places the guide's own output on the ladder. Reading it is a useful shock, because the great majority of what Part II produced sits on the third rung, and the chapters that produced it were the most technically demanding in the book.

Table 11.6

The evidence hierarchy, with this guide's own output placed on it

Rung	What it is	Where it appears in this guide	The strongest sentence it licenses
1. Anecdote	A case, a quotation, a memorable single observation	The merchant's conviction (Section 10.1); "customers keep telling us…"	"Here is something that happened, which may be worth investigating."
2. Description and association	Rates, cross-tabs, correlations measured in observed data	Chapters 5 and 6; the correlations of Section 7.3; the 5.30-point difference of Section 11.1	"These things occur together in our data, by this much."
3. Model-based association and prediction	Multivariable estimates with controls; out-of-sample predictive validation; forecasts	Chapters 7, 8, 9, and 10; the adjusted 1.80-point difference of Section 11.3	"Holding these measured factors constant, X accompanies Y by this much," or "this model predicts Y this accurately on data it has not seen."
4. Quasi-experiment	Pre/post with a comparison group; difference-in-differences and its relatives	Section 11.9	"Under the stated parallel-trends assumption, the treatment appears to have changed the outcome by this much."
5. Randomized experiment, competently executed	Random assignment to treatment and control, analyzed as assigned, with assignment integrity documented	Sections 11.4 through 11.7; the ab_test deliverable	"Among the tested population, in the tested window, at the tested dose, the treatment caused a change of this much, within this interval."

Three readings of the table matter more than the table itself. The first is that the top rung is narrow. Even a clean experiment licenses a sentence that is heavily qualified by population, window, and dose, and the qualifications are not throat-clearing — they are the boundary of what was tested, and Section 11.13 will show what happens when they are dropped in the retelling. The second is that the lower rungs are not the losers. Chapter 6's segments could not have been discovered by an experiment; Chapter 9's churn model does a job no experiment does, which is ranking individuals for action; Chapter 10's forecast answers a question about the future that no experiment addresses. The hierarchy ranks causal warrant, not usefulness, and an organization that ran only experiments would be blind. The third and most practical is that the hierarchy is a vocabulary constraint that can be checked in a draft. Verbs like *drives*, *causes*, *lifts*, *generates*, and *produces* are rung-five verbs; *is associated with*, *accompanies*,

predicts, and *is higher among* are rung-two and rung-three verbs. An analyst editing a deck for these words, once, before it ships, will catch nearly every overclaim she is about to make.

CONCEPT

The Ladder Is a Vocabulary Constraint

The practical discipline this chapter leaves behind outlives its arithmetic, and it is small enough to apply in a hallway. For any sentence about to be spoken to a decision-maker, ask which rung produced it, then check whether the verb belongs to that rung. "The loyalty program drives a 22 percent higher repeat rate" is a rung-five sentence resting on rung-two evidence, and the fix is not to hedge it into meaninglessness but to say the true thing: "Loyalty members repeat 22 percent more often than non-members; we have not tested whether enrolling a customer causes that, and the customers who enroll are already our most frequent."

Stakeholders do not resent the second sentence. They resent discovering, two quarters into a program built on the first one, that nobody had ever checked. The analyst who speaks in rung-appropriate verbs from the beginning is the one whose rung-five sentences get believed when she finally has one.

Source: Course concept developed for this guide, informed by Angrist and Pischke (2015).

11.11 AI as an Experimentation Assistant

The division of labor the guide has refined since Chapter 7 — the assistant drafts mechanics, the analyst audits meaning — arrives in this chapter at a task where the assistant's competence and the task's requirements are almost perfectly misaligned, and the misalignment is worth naming before the failure modes are listed. An experiment's arithmetic is trivial and its epistemology is everything. Assistants are excellent at arithmetic and structurally incapable of the epistemology, because the epistemology lives in facts that are nowhere in the data: whether the assignment was random, whether the metric was declared before the outcomes were seen, whether this is the first look or the fourteenth, whether the subgroup in the prompt was chosen before or after somebody noticed it was winning. Hand an assistant two columns and ask whether the difference is significant and it will answer — correctly, fluently, with an interval and a p-value and a sentence about the result being statistically significant — and it will answer in exactly the same way whether the data came from Section 11.4's randomized design or from Section 11.1's targeted send, whether the test was declared or discovered, and whether it is being run for the first time or the fortieth. Nothing in the numbers reveals the difference, and nothing in the assistant's training makes it ask.

The experimentation-specific failure modes follow from that gap, and they are named here for the audit. The observational file analyzed as a trial: A/B machinery applied to data in which somebody chose who was treated, producing a confident causal sentence about the selection bias of Section 11.3 — this is the highest-frequency and highest-cost failure, and it is invited by the request itself whenever the prompt says "campaign" instead of "experiment." The uncomplaining test on peeked data: the assistant computes significance for whatever window it is handed and will not ask when the test started, how many times it has been examined, or whether the analyst is stopping now because of what she just saw. The invented design: asked to design a test, the assistant fills every gap the prompt left — inventing a metric definition, choosing a duration, assuming a unit of randomization, assuming a treatment dose, sometimes assuming an eligibility rule — and does so silently and plausibly, so that the resulting registration reads like a plan the analyst made. The per-protocol drift: asked to analyze an experiment with delivery failures, the assistant will cheerfully drop the undelivered customers and never mention that it has just replaced the intention-to-treat comparison with an observational one. The segment harvest: asked to "check whether it worked for different customer groups," the assistant will happily test twenty subgroups and report the significant ones, executing Section 11.8's multiple comparisons problem at machine speed with no note that the p-values are meaningless as printed. The lift ambiguity: relative and absolute lift conflated or converted incorrectly, relative lift reported without the baseline, and a ratio interval produced by dividing a difference interval by the control rate,

which is Section 11.6's shortcut committed by autocomplete. The naked winner: results reported as "treatment wins" with no interval, no practical-significance line, no stated dose, and no statement of the tested population — the assistant inherits the internet's preference for decisive conclusions. The grain slip: data randomized by customer but analyzed by order or session, inflating the apparent sample size and shrinking the interval to a width the design never earned. And the power omission: a null result narrated as "no effect" rather than "this test could not have detected an effect smaller than X," which is the Section 11.5 distinction the assistant will not make unless the prompt makes it.

Where the assistant genuinely helps, the list is long and worth using deliberately, because most of an experiment's labor is mechanical. Sample-size and power arithmetic is fully specifiable and the assistant is fast and accurate at it, including the sensitivity table of required sample size across a range of minimum detectable effects that makes the inverse-square relationship concrete for a planning meeting. Design checklists drafted against Table 11.3 catch omissions a human drafter skips. Balance-check, flow-table, and sham-split code is boilerplate. The two-proportion analysis, its two intervals, and the peeking simulation of Lab 11.2 are exactly the kind of well-specified mechanical work Section 8.9's frozen-rules discipline makes safe to delegate. Break-even arithmetic and the sensitivity band across contribution and cost are pure computation. And an assistant is genuinely useful as an adversary: asked to argue against a result — to list every rival explanation for an observed difference, every contamination path, every way the tested population or dose differs from the rollout's — it produces a better checklist than most analysts generate unaided, and the exercise costs nothing.

The governing instrument extends the audit lineage once more. Table 8.5's five points run unchanged — frame, leakage, split hygiene, baseline, error in decision units — with "baseline" reading, in this chapter, as the control arm and the break-even line. Table 11.7 adds the three checks experimentation makes necessary, and its first point is the one that has no analogue anywhere else in the guide, because it is a check on chronology rather than on code.

Table 11.7

The experimentation supplement to the five-point audit (run with Table 8.5)

Supplemental point	The check	Fails when
E1. Registration precedes analysis	A dated preregistration exists (Table 11.3), and the metric, arms, dose, unit, sample size, duration, decision rule, and stopping rule in the analysis match it exactly; deviations are listed as deviations	The metric or window in the analysis differs from the plan; no plan exists; the "primary" metric was chosen after the results were seen; the test stopped at an unplanned moment; the dose is nowhere stated
E2. Assignment integrity	Assignment was made by a stated chance mechanism independent of customer characteristics; the assignment-flow record of Table 11.4 is reported from randomization through outcome observation; the analysis is intention-to-treat; covariate balance is read as a diagnostic rather than a test; the analysis grain equals the assignment grain; contamination paths are named with expected direction	Anyone chose who was treated; undelivered or noncompliant customers were dropped from the primary comparison; a covariate imbalance is used to discard a valid randomization; sessions or orders are analyzed after customers were randomized
E3. Inference honesty	One primary metric analyzed once, with everything else labeled exploratory; absolute lift reported with the baseline rate beside it; a difference interval and a separately constructed ratio interval; the interval compared against a break-even line computed in advance from a stated dose, with its sensitivities; null results reported with their minimum detectable effect	Significance appears without an interval; a subgroup is the headline; relative lift is quoted alone or its interval is a rescaled difference interval; "no effect" is claimed by an underpowered test

AI IN PRACTICE

Preregister With the Assistant, Then Audit the Analysis Against the Registration

The two-prompt pattern of Sections 8.11, 9.11, and 10.11, restructured around this chapter's chronology, because here the order of the prompts is itself the control. Step one, before any data exists, is the registration prompt: "I am testing whether a twelve-hour early-access SMS lifts fourteen-day conversion among SMS-opted-in customers. Draft a preregistration covering every row of the following template — decision, primary metric with its four elements, arms including what control receives, treatment dose in number of exposures, assignment mechanism, unit of randomization, eligible population, sample size with the power calculation shown, duration, planned analysis, decision rule, stopping rule, secondary metrics labeled exploratory, and guardrails that are computable from the columns I have listed. Where the information I gave you is insufficient to fill a row, write UNSPECIFIED and ask me; do not fill it yourself." The final clause is the whole prompt, because the invented-design failure mode is defeated only by making the assistant's gaps visible.

Step two, the prediction, before the analysis runs: write down the lift you expect and the interval width you expect, computing the latter from the arm sizes and the baseline rate. The analyst who has done the power calculation already knows the interval will be roughly two points wide, and an interval that comes back at a tenth of that width means the grain slipped. Step three, the analysis prompt, which begins by pasting the registration back in: "Here is the registration. Analyze the attached file exactly as registered — intention-to-treat, no other metrics, no subgroups, one analysis. Report the assignment-flow counts, the control rate, the treatment rate, absolute lift in percentage points with a 95 percent interval on the difference, relative lift with a 95 percent interval on the ratio computed as a ratio, and the p-value. Do not interpret."

Step four, the audit: run Table 8.5's five points, then Table 11.7's three, line by line, and grade your prediction. Then the adversarial prompt, which is where the assistant earns its keep: "List every reason this estimate might not transfer to a chain-wide quarter-long rollout at two alerts per week instead of one, and every contamination path you can construct." Harvest that list into the deliverable's limitations. Only after all of it, the drafting prompt: "Write the rollout recommendation paragraph using only these numbers." Audit its verbs against Section 11.10's ladder — an experiment licenses *caused, among the tested population, at the tested dose, within this interval*; it never licenses *will* — file both exchanges per Appendix D, and sign the recommendation as the analyst of record.

Source: Course concept developed for this guide, informed by Kohavi et al. (2020) and Nosek et al. (2018).

11.12 Hands-On Application in Python and Google Colab

The preceding sections built the argument; this section runs it on StyleCraft's own files, in two labs that follow the chapter's chronology rather than its section order. Lab 11.1 establishes credibility before any effect is estimated: the observational trap reconstructed by hand at miniature scale so that the 5.30 points decomposes in front of the student, then the experiment's assignment audited for integrity with a data contract, an assignment-flow table, balance profiling, and a sham-split analysis-path check. Lab 11.2 sizes, estimates, and prices the effect: the power calculation that justified the sample, the registered intention-to-treat analysis with both of its intervals, checked against the known true lift the dataset ships; the deliberate peeking simulation in which a false winner is manufactured from data containing no effect at all; the practical-significance comparison against the break-even line with its sensitivity band; a temporal-stability diagnostic that is carefully not a novelty test; and the round trip through the assistant that produces the rollout paragraph.

The labs use statsmodels throughout for the proportion tests, the interval constructions, and the power calculations (Seabold & Perktold, 2010); the three functions they depend on are documented by the statsmodels developers (2026a, 2026b). They use the certified ab_test file, whose columns follow the course data dictionary: one row per assigned customer, with customer_id, assign_date, variant (control or treatment), converted (1 if the customer placed at least one order within fourteen days of assignment), order_revenue for the same window, and the pre-assignment characteristics home_metro, generation, loyalty_tier, app_user, prior_orders_90d, and prior_spend_90d. Two conventions hold across every cell. Code cells are numbered continuously across both labs, so that the deliverable and the exercises can cite them. And every output is predicted before it is computed, per the standing rule. AI assistants may draft any code cell (Appendix C has templates; Appendix A covers Colab mechanics), and every exchange is documented per Appendix D.

11.12.1 Lab 11.1, Part A: The Observational Trap, Decomposed by Hand

The miniature this chapter needs is not a sample of rows but a table of four cells, because selection bias is a fact about composition and four numbers are enough to show it completely. The cells are the observational record of Table 11.1: sixteen thousand customers, crossed by whether they received the Drop Alert and by whether they were engaged at the time of the send. Before running anything, predict two quantities and write them down — the overall difference between the received and not-received groups, and the difference within each engagement stratum — and predict which of the two the CMO's deck reported.

Code 11.1. Aggregate the observational comparison

```
import pandas as pd

obs = pd.DataFrame({
    "engaged":     ["Engaged", "Engaged",
                    "Not engaged", "Not engaged"],
    "drop_alert":  ["Received", "Not received",
                    "Received", "Not received"],
    "customers":   [6000, 2000, 2000, 6000],
    "conversions": [708,  200,  96,   180],
})
obs["rate"] = obs["conversions"] / obs["customers"]

naive = (obs.groupby("drop_alert")[["customers", "conversions"]]
            .sum())
naive["rate"] = naive["conversions"] / naive["customers"]
print(naive)

recv = naive.loc["Received", "rate"]
notr = naive.loc["Not received", "rate"]
print(f"naive difference: {(recv - notr) * 100:.2f} pp")
print(f"relative:         {recv / notr - 1:+.1%}")
```

Expected output: the two-row aggregate table with customers, conversions, and rate for the received and not-received groups, then the naive difference in percentage points and the same difference expressed relatively.

Input is the crossed record; the transformation is a sum that discards the engagement dimension; the output is the number that started the argument, and the discarding is the point. Note that the four literal cells are the chapter's own Table 11.1, entered by hand so that every figure downstream can be checked with a calculator.

Code 11.2. Standardize by engagement composition

```
wide = obs.pivot(index="engaged", columns="drop_alert",
                 values="rate")
wide["within_diff_pp"] = ((wide["Received"]
                           - wide["Not received"]) * 100)
print(wide.round(4))
```

```
# What the treated group would have converted at with no
# campaign: its own engagement mix, at untreated rates.
# This is an ADJUSTED difference under the assumption that
# engagement is the only confounder -- not a causal effect.
w = (obs.loc[obs["drop_alert"] == "Received"]
        .set_index("engaged")["customers"])
w = w / w.sum()
counterfactual = (w * wide["Not received"]).sum()

print(f"treated actual:         {recv:.2%}")
print(f"treated counterfactual: {counterfactual:.2%}")
print(f"adjusted difference (pp): "
      f"{(recv - counterfactual) * 100:.2f}")
print(f"composition component (pp): "
      f"{(counterfactual - notr) * 100:.2f}")
```

Expected output: the stratified table with each arm's rate side by side and the within-stratum difference in percentage points, then the treated group's actual rate, its constructed counterfactual rate, and the split of the naive difference into an adjusted difference and a composition component.

The second cell does the work of Section 11.3 in a dozen lines. The pivot puts the two rates side by side within each engagement stratum; the weights w are the treated group's own engagement mix; and applying that mix to the untreated rates constructs the rate the treated group would have shown had the campaign never existed. Read the print labels carefully, because they are the correction this chapter's draft needed: the cell prints an adjusted difference and a composition component, not an effect and a bias. Calling the first quantity the campaign's effect would smuggle in four assumptions the record cannot check, and the experiment of Lab 11.2 is what makes those assumptions unnecessary rather than merely plausible.

VERIFICATION CHECK

Before you run: compute all of it by hand. The stratum rates are 708 ÷ 6,000 = 11.80 percent, 200 ÷ 2,000 = 10.00 percent, 96 ÷ 2,000 = 4.80 percent, and 180 ÷ 6,000 = 3.00 percent, so the within-stratum differences are 1.80 points and 1.80 points — identical, which is the designed signature of a record in which the treatment did the same thing to both kinds of customer. The aggregates are 804 ÷ 8,000 = 10.05 percent and 380 ÷ 8,000 = 4.75 percent, a naive difference of 5.30 points and a relative lift of 111.6 percent. The counterfactual is 0.75 × 10.00 + 0.25 × 3.00 = 8.25 percent, so the adjusted difference is 10.05 − 8.25 = 1.80 points and the composition component is 8.25 − 4.75 = 3.50 points.

After you run: confirm that the two components sum exactly to the naive 5.30, because the additivity is what makes the decomposition a decomposition rather than an adjustment. Then verify the composition component a second way, from mix alone: the treated group was 75 percent engaged and the untreated group 25 percent, a 50-point difference in mix, and engaged customers converted 7.00 points higher untreated (10.00 against 3.00), so 0.50 × 7.00 = 3.50 points arrive from nothing but who was chosen. Now write the sentence the miniature has earned, in the finance partner's language rather than the analyst's: two-thirds of the campaign's apparent performance was the CRM team's targeting skill, correctly measured and incorrectly attributed. Then write the second sentence, which is the one this chapter cares about more: the remaining 1.80 points is an adjusted observational difference that equals the campaign's effect only if engagement is the sole confounder, measured without error, and the only thing distinguishing the two groups — and the only reason the adjustment was possible at all is that engagement happened to be recorded, which is a piece of luck the next confounder will not repeat.

> Investigate if: your hand arithmetic disagrees with the printed figures anywhere. Every number in these two cells is reproducible with a calculator, so a mismatch is arithmetic rather than convention.

11.12.2 Lab 11.1, Part B: Assignment Integrity — the Data Contract, the Flow Record, Balance, and the Sham Split

The experiment's credibility rests entirely on the claim that a chance mechanism, and nothing else, decided who received the Drop Alert. This part audits that claim before a single effect is estimated, which is the order the deliverable will also follow: an experiment whose assignment cannot be defended does not get to report a lift. Four cells do the work, and the first of them is a contract rather than an analysis.

Code 11.3. Load and validate the experimental file

```
import numpy as np
import pandas as pd

ab = pd.read_csv("ab_test.csv", parse_dates=["assign_date"])

# THE DATA CONTRACT. Every assertion is a claim the analysis
# depends on. A failure stops the lab; it is not repaired
# silently, and it is not repaired by dropping rows.
required = ["customer_id", "assign_date", "variant",
            "converted"]
missing = [c for c in required if c not in ab.columns]
assert not missing, f"columns absent from the file: {missing}"
assert ab[required].notna().all().all(), (
    "a required field is null: trace the source before "
    "computing anything")
assert ab["customer_id"].is_unique, (
    "customer_id repeats: this may be a duplicated RECORD or "
    "a customer randomized twice. Trace which before acting")
assert set(ab["variant"]) == {"control", "treatment"}, (
    f"unexpected arm labels: {sorted(set(ab['variant']))}")
assert ab["converted"].isin([0, 1]).all(), (
    "converted is not a clean 0/1 flag")

print("rows:", len(ab), " columns:", len(ab.columns))
print("assignment window:",
      ab["assign_date"].min().date(), "to",
      ab["assign_date"].max().date())
print(ab["variant"].value_counts().to_string())
```

Expected output: the row and column counts, the first and last assignment dates, and the count of customers in each arm; five assertions pass silently.

Input is the certified file; the transformation is nothing but validation; the output is permission to proceed. Read the duplicate assertion's message rather than just its condition, because it is a correction to the draft of this chapter. A repeated customer_id does not establish that a customer was randomized twice. It may equally be one randomization written to the file twice — a union join instead of a unique key, per Chapter 4's join disciplines — and the two defects have different consequences and different repairs. The code therefore stops and instructs the analyst to trace the source, rather than asserting a causal interpretation of a data defect it cannot distinguish.

Code 11.4. Build the assignment-flow table

```
# The flow record of Table 11.4. Randomized customers stay in
# the arm they were assigned to (intention-to-treat); nothing
# here is a filter, and delivery is reported, never applied.
PLANNED = {"control": 5000, "treatment": 5000}
```

```
flow = pd.DataFrame(index=["randomized",
                           "primary outcome observed",
                           "included in ITT analysis"])
for arm in ("treatment", "control"):
    g = ab.loc[ab["variant"] == arm]
    flow[arm] = [len(g),
                 int(g["converted"].notna().sum()),
                 int(g["converted"].notna().sum())]
print(flow.to_string())

for arm, n in PLANNED.items():
    got = int((ab["variant"] == arm).sum())
    flag = "OK" if got == n else "INVESTIGATE"
    print(f"{arm:<10s} planned {n:>5d}  realized {got:>5d}"
          f"  {flag}")

print("\nUnder complete randomization the realized counts "
      "should equal\nthe planned counts exactly. A shortfall "
      "is post-assignment loss:\nreport it, trace it, and do "
      "NOT drop the affected customers.")
```

Expected output: a three-row flow table giving the randomized, outcome-observed, and analyzed counts for each arm; one planned-versus-realized line per arm with an OK or INVESTIGATE flag; and the standing note on what a shortfall means.

Input is the validated file and the registration's declared allocation; the transformation is counting at three stages; the output is page two of the deliverable. The delivery row of Table 11.4 is absent from this cell for an honest reason: the certified ab_test file does not carry a delivery flag, and inventing one would teach students to fabricate the very evidence the flow record exists to demand. Where a messaging log is available, it joins here as an additional reported row — and it stays a reported row. The final print is the cell's real payload, because the instinct a delivery shortfall provokes is exactly the wrong one, and the code says so at the moment the instinct arrives.

Code 11.5. Profile pre-assignment balance

```
cats = ["home_metro", "generation", "loyalty_tier",
        "app_user"]
for col in cats:
    # Represent missingness explicitly. crosstab drops NaN
    # silently, which would hide an imbalance in whether a
    # characteristic was recorded at all.
    profiled = ab[col].astype("string").fillna("Missing")
    share = pd.crosstab(profiled, ab["variant"],
                        normalize="columns")
    share["diff_pp"] = ((share["treatment"]
                         - share["control"]) * 100)
    print(f"\n{col}")
    print(share.round(4).to_string())

nums = ["prior_orders_90d", "prior_spend_90d"]
print("\n", ab.groupby("variant")[nums]
              .agg(["mean", "std", "count"]).round(3))

print("\nRead these as DIAGNOSTICS, not as tests. Chance "
      "imbalance occurs\nunder a valid procedure; a "
      "conspicuous imbalance triggers an audit\nof the "
      "assignment code, not a re-randomization.")
```

Expected output: one composition table per categorical characteristic, each with the treatment and control shares, an explicit Missing row where any values are absent, and the difference in percentage points; then a table of means, standard deviations, and counts for the two numeric characteristics; then the standing note on how to read them.

The categorical block compares each arm's composition share by share; the numeric block compares means and spreads. Every variable used here is measured before assignment, which is what makes the check meaningful — a "balance check" on a post-assignment variable is a treatment effect wearing a diagnostic's costume. Two details are deliberate. Missing values are converted to an explicit "Missing" category rather than being dropped by crosstab, because whether a characteristic was recorded at all can itself differ between arms and is exactly the kind of structured defect a broken randomizer produces. And the closing note is printed inside the cell so that it travels with the output into whatever document the student pastes it into.

VERIFICATION CHECK

Before you run: predict a shape rather than a number. Every categorical share should differ between arms by a fraction of a percentage point, the differences should scatter in both directions with no pattern, and the numeric means should agree closely with standard deviations that match. Predict the realized arm sizes against the registration's declared 5,000 and 5,000, and predict how many "Missing" rows you expect to see.

After you run: check the flow table and the balance profile together, and record the pair as the assignment certificate. It is page two of the deliverable, and Section 11.13 will not let you skip it. Then reason through the three findings that would change what you do next, and note that only one of them stops the lab. Arms of materially unequal size under complete randomization mean customers were lost after assignment, which must be traced and reported — and the affected customers stay in the intention-to-treat comparison while you trace it. A single covariate imbalanced far beyond the others, especially one related to geography or signup date, is the fingerprint of a randomizer keyed to something structured rather than random, and it sends you to the assignment code and its log. And a prior_spend_90d difference of a percent or two between arms is what a chance mechanism produces on ten thousand customers; it is not grounds for discarding the comparison, and if you want the precision back, the instrument is a covariate-adjusted analysis declared in advance, not a re-randomization.

Investigate if: any covariate's difference is large enough that you would have noticed it without looking for it, or if the Missing share differs between arms. The first is an audit of the assignment mechanism. The second is an audit of the data pipeline, and it matters even when the outcome looks fine, because a characteristic recorded differently across arms usually means something else was too.

Code 11.6. Run the sham-split analysis-path check

```
from statsmodels.stats.proportion import proportions_ztest

# NOT an A/A test. This splits COMPLETED control-arm outcomes
# into two sham groups, so it grades the analysis path only:
# the metric, the test statistic, and the code. It touches
# neither the randomizer nor delivery nor tracking.
rng = np.random.default_rng(2026)
ctrl = ab.loc[ab["variant"] == "control",
              "converted"].to_numpy()

for seed in range(5):
    g = np.random.default_rng(2026 + seed)
    sham = g.integers(0, 2, size=ctrl.size)
```

```
    counts = np.array([ctrl[sham == 1].sum(),
                       ctrl[sham == 0].sum()])
    sizes = np.array([(sham == 1).sum(), (sham == 0).sum()])
    rates = counts / sizes
    stat, pval = proportions_ztest(counts, sizes)
    print(f"seed {2026 + seed}  sizes {sizes}"
          f"  diff {(rates[0] - rates[1]) * 100:+.2f} pp"
          f"  z {stat:+.2f}  p {pval:.3f}")
```

Expected output: five lines, one per seed, each giving the two sham arm sizes, the difference between their conversion rates in percentage points, the test statistic, and the p-value.

The cell splits the control arm at random into two halves that received identical treatment, then runs the same test the real analysis will run. The true effect here is exactly zero by construction, so the cell grades the analysis path rather than the campaign — and the comment at the top says precisely which parts of the apparatus it does and does not grade, which is the distinction Section 11.5 draws and the reason this guide no longer calls the procedure an A/A test. A genuine A/A test would push both sham arms through the production randomizer, the production send, and the production tracking, and StyleCraft's CRM team owes one before the next experiment rather than after it.

VERIFICATION CHECK

Before you run: predict the output. Each difference should land within a few tenths of a percentage point of zero, and each p-value should be an arbitrary number between 0 and 1 with no particular tendency to be small. Predict how many of the five seeds you expect to return a p-value below 0.05, and write the number down before you look.

After you run: record all five p-values. The lesson is in the expected count rather than in any single line: at a 5 percent threshold, roughly one run in twenty will report a "significant" difference between two groups that received identical treatment, and when it does, nothing in the output will look wrong. That is why a single significant result here — or in a genuine A/A test — is an occasion to investigate rather than a verdict that the infrastructure is broken, and why repeated or systematic failures are the signal that matters. Write that sentence down, because it is the entire intuition behind Section 11.8's peeking and multiple-comparisons failures, and Code 11.10 will turn it into a machine.

Investigate if: the sham arm sizes are wildly unequal, or several of five seeds return small p-values. The first means the sham flag is not doing what you think it is. The second, on a known-zero effect, means the metric or the test call is wrong — which is exactly the defect this cell exists to catch, and catching it here is far cheaper than catching it in the primary analysis.

11.12.3 Lab 11.2, Part A: Sizing the Test That Was Already Run

Power is an owned concept of this chapter and a stated learning objective, and a chapter that asserts a sample-size calculation without running one is teaching a habit it does not model. This part therefore opens where the CRM team opened, before any outcome existed: with the arithmetic that turned a business judgment about the smallest lift worth detecting into a number of customers per arm. Predict the answer first. At a 6 percent baseline, a 1.5-point minimum detectable effect, a 5 percent significance level, and an 80 percent power target, write down how many customers per arm you think the calculation will demand.

Code 11.7. Calculate required sample size and power

```
import numpy as np
from statsmodels.stats.power import NormalIndPower
from statsmodels.stats.proportion import proportion_effectsize

BASELINE = 0.060      # control conversion, from history
```

```
MDE      = 0.015      # smallest lift worth detecting: a
                      # BUSINESS judgment, not a statistic
ALPHA    = 0.05
POWER    = 0.80

power_solver = NormalIndPower()

def n_per_arm(baseline, mde, alpha=ALPHA, power=POWER):
    es = proportion_effectsize(baseline + mde, baseline)
    return power_solver.solve_power(
        effect_size=es, alpha=alpha, power=power,
        ratio=1.0, alternative="two-sided")

def power_at(baseline, mde, n, alpha=ALPHA):
    es = proportion_effectsize(baseline + mde, baseline)
    return power_solver.solve_power(
        effect_size=es, alpha=alpha, power=None,
        nobs1=n, ratio=1.0, alternative="two-sided")

need = n_per_arm(BASELINE, MDE)
print(f"required customers per arm: {int(np.ceil(need))}")
print(f"registered per arm:         5000")

print("\nsensitivity: sample size and realized power at 5,000")
print(f"{'MDE (pp)':>9}{'n per arm':>12}{'power @5,000':>14}")
for mde in (0.005, 0.010, 0.015, 0.018):
    print(f"{mde * 100:>9.1f}"
          f"{int(np.ceil(n_per_arm(BASELINE, mde))):>12d}"
          f"{power_at(BASELINE, mde, 5000):>14.3f}")

print("\npower for the planted 1.8-point lift, by arm size")
for n in (1000, 2500, 5000):
    print(f"  n = {n:>5d} per arm -> power "
          f"{power_at(BASELINE, 0.018, n):.3f}")
```

Expected output: the required customers per arm beside the registered 5,000; a four-row sensitivity table giving the required sample size and the realized power at 5,000 per arm for minimum detectable effects of 0.5, 1.0, 1.5, and 1.8 percentage points; and the realized power for the planted 1.8-point lift at 1,000, 2,500, and 5,000 per arm.

Input is four declared numbers, three of them conventions and one of them a business judgment. The transformation is the standard sample-size calculation for a difference in two proportions. The output is the justification for the sample-size row of Table 11.3, plus the sensitivity table that makes the inverse-square relationship visible instead of asserted. Note the structure of the two helper functions: they differ only in which argument is left as None, because a power calculation and a sample-size calculation are the same equation solved for different unknowns, and seeing that in code is worth more than memorizing either formula.

VERIFICATION CHECK

Before you run: predict the required sample size for the registered 1.5-point minimum detectable effect, and predict the shape of the sensitivity table — specifically, predict the ratio between the sample size needed for a 0.5-point effect and the sample size needed for a 1.5-point effect, using the inverse-square rule of thumb from Section 11.5. Then predict the power at 1,000 per arm for the planted 1.8-point lift, and state in one sentence what a value near one-third would mean for a team that ran that test and reported no effect.

After you run: reconcile the figures. The calculation returns 4,379 customers per arm, which the team rounded to 5,000; the 0.5-point row demands 36,778 per arm, roughly eight and a half times the 1.5-point requirement, which is the inverse-square rule showing up slightly softened by the fact that the baseline moves too. At 5,000 per arm the test has 94.5 percent power against the planted 1.8-point lift and 84.9 percent against the 1.5-point effect it was sized for, and only 35.6 percent power at 1,000 per arm. Write the sentence that last number earns: a team running this experiment at a thousand customers per arm would have had a two-in-three chance of missing a real, profitable effect entirely and reporting that the Drop Alert does not work. Then write the second sentence, which is the ethical one: an underpowered experiment still assigns real customers to a control arm, and it buys nothing with their inconvenience.

Investigate if: your required sample size differs from 4,379 by more than a customer or two. The likely cause is a different effect-size convention — some calculators use the raw difference in proportions rather than the arcsine-transformed effect size that proportion_effectsize computes — and the fix is to state which convention you used rather than to assume the other one was wrong.

11.12.4 Lab 11.2, Part B: The Registered Analysis and the Known True Lift

The assignment is certified and the sample is justified. The analysis may now run. It runs exactly once, exactly as registered, at the grain of assignment, on every randomized customer with an observed outcome.

Code 11.8. Estimate absolute and relative lift

```
from statsmodels.stats.proportion import proportions_ztest

# Intention-to-treat: grouped by ASSIGNED variant, with no
# post-assignment filter of any kind applied above this line.
s = (ab.groupby("variant")["converted"]
       .agg(customers="size", conversions="sum"))
s["rate"] = s["conversions"] / s["customers"]
print(s.to_string())

c_n = int(s.loc["control", "customers"])
c_x = int(s.loc["control", "conversions"])
t_n = int(s.loc["treatment", "customers"])
t_x = int(s.loc["treatment", "conversions"])
c_rate, t_rate = c_x / c_n, t_x / t_n

abs_lift = t_rate - c_rate
rel_lift = abs_lift / c_rate
stat, pval = proportions_ztest([t_x, c_x], [t_n, c_n])

print(f"\ncontrol:   {c_x:>5d} / {c_n:>5d} = {c_rate:.4%}")
print(f"treatment: {t_x:>5d} / {t_n:>5d} = {t_rate:.4%}")
print(f"absolute lift: {abs_lift * 100:+.2f} pp")
print(f"relative lift: {rel_lift:+.1%}")
print(f"z = {stat:.2f}   two-sided p = {pval:.4f}")
```

Expected output: the two-row arm summary with customers, conversions, and rate; then the control and treatment rates stated as fractions and percentages, the absolute lift in percentage points, the relative lift as a percentage, and the test statistic with its two-sided p-value.

Input is the certified file at the grain of assignment; the transformation is two proportions and their difference; the output is the estimate and its p-value. Note what the cell does not contain: no subgroups, no secondary metrics,

no loop over anything, and no filter on delivery or engagement. That absence is the registration being honored, and it is the most important feature of the code.

Code 11.9. Compute difference and ratio intervals

```
from statsmodels.stats.proportion import (
    confint_proportions_2indep)

# The PRIMARY result: an interval on the DIFFERENCE.
d_lo, d_hi = confint_proportions_2indep(
    t_x, t_n, c_x, c_n, compare="diff", method="wald")
print(f"absolute lift {abs_lift * 100:+.2f} pp"
      f"   95% CI [{d_lo * 100:+.2f}, {d_hi * 100:+.2f}]")

# The relative figure needs its OWN interval, computed as a
# ratio. Rescaling the difference interval by the observed
# control rate treats an estimate as a known constant.
r_lo, r_hi = confint_proportions_2indep(
    t_x, t_n, c_x, c_n, compare="ratio", method="log")
print(f"risk ratio    {t_rate / c_rate:.3f}"
      f"        95% CI [{r_lo:.3f}, {r_hi:.3f}]")
print(f"relative lift {rel_lift:+.1%}"
      f"      95% CI [{r_lo - 1:+.1%}, {r_hi - 1:+.1%}]")

# The shortcut, printed only so its error is visible.
print(f"\nWRONG (difference CI rescaled by control rate): "
      f"[{d_lo / c_rate:+.1%}, {d_hi / c_rate:+.1%}]")

# Both standard errors, for the record.
se_un = np.sqrt(c_rate * (1 - c_rate) / c_n
                + t_rate * (1 - t_rate) / t_n)
pool = (c_x + t_x) / (c_n + t_n)
se_pool = np.sqrt(pool * (1 - pool) * (1 / c_n + 1 / t_n))
print(f"SE unpooled {se_un * 100:.4f} pp"
      f"   SE pooled {se_pool * 100:.4f} pp")
```

Expected output: the absolute lift with its 95 percent interval on the difference; the risk ratio with its 95 percent interval; the relative lift with the interval implied by the ratio interval; the interval the rescaling shortcut would have produced, labeled as wrong; and the unpooled and pooled standard errors.

Input is the four counts from Code 11.8; the transformation is two different interval constructions on the same data; the output is the pair of intervals the deliverable reports and the one it must not. The ratio interval is computed on the log scale and exponentiated, which is the standard construction and the reason the interval is asymmetric around the point estimate. Printing the shortcut beside the honest answer is deliberate pedagogy rather than clutter: students who have never seen the two side by side tend to assume the difference is enormous, and the more useful lesson is that it is modest here and unpredictable in general — which is exactly why the shortcut cannot be trusted rather than merely tolerated.

VERIFICATION CHECK

Before you run: predict the numbers in full, because this is the chapter's verification theme. The control rate should land near 6.0 percent, since that is the baseline the power calculation was built on; the treatment rate near 7.8 percent; the absolute lift near 1.8 percentage points, because the dataset's published design note states the planted true lift and the whole purpose of a correct analysis is to recover it. Predict the interval's width too, from the arm sizes: with 5,000 per arm at rates near 6 and 8 percent, the standard error of the difference is close to 0.51 percentage points, so the 95 percent interval is roughly two points wide. Predict where the relative-lift interval will fall, and predict whether the rescaling shortcut will look too wide or too narrow.

After you run: grade four things. Does the point estimate land near 1.8? Does the interval contain 1.8? Is the interval about as wide as you predicted — an interval far narrower than two points is the grain slip of Section 11.11, so confirm the analysis counted customers rather than orders, and an interval far wider means the arms are smaller than the registration promised, which sends you back to Code 11.4. And is the relative interval, running from roughly 12 to 50 percent, visibly wider than the 13-to-47 the shortcut produced? Then write the two sentences the deliverable opens with: the control and treatment rates stated plainly, and the lift stated in percentage points with its interval attached and the relative figure second. Finally, compare the interval against the observational 5.30 points from Part A and state the comparison precisely, because the draft of this chapter stated it wrongly: the observational difference is nearly three times the experimental point estimate of 1.80 points and nearly twice the upper end of the experimental interval. Those are two different multiples of two different quantities, and quoting the larger one against the smaller reference is the kind of small inaccuracy that a finance partner will find.

Investigate if: the point estimate is far from 1.8, or the interval excludes it. Neither should happen on the certified file, and both are findings about the pipeline rather than about StyleCraft. Check first that the grouping variable is the assigned variant and not something derived from behavior after assignment, then that no filter was applied between Code 11.3 and Code 11.8.

11.12.5 Lab 11.2, Part C: Manufacturing a False Winner

Section 11.8 asserted that peeking inflates false positives far beyond the nominal 5 percent, and gave two numbers rather than one. This part makes the student produce both, on StyleCraft's real data, in a setting where the true effect is known to be exactly zero because both sham groups come from the control arm.

Code 11.10. Simulate repeated peeking

```
import numpy as np
from statsmodels.stats.proportion import proportions_ztest

ctrl = ab.loc[ab["variant"] == "control",
              "converted"].to_numpy()
LOOKS, ALPHA, TRIALS, MIN_PER_ARM = 14, 0.05, 1000, 100

def one_trial(y, g, keep_path=False):
    """One sham experiment examined LOOKS times. Returns the
    verdict under three stopping policies -- final (look once
    at close), any (stop at any significant difference), and
    favor (stop only when significant AND treatment ahead) --
    plus, optionally, the p-value at every look."""
    order = g.permutation(y.size)
    y, arm = y[order], g.integers(0, 2, size=y.size)
```

```
    cuts = np.linspace(y.size / LOOKS, y.size,
                       LOOKS).astype(int)
    final = any_hit = favor_hit = False
    path = []
    for k in cuts:
        a, b = y[:k][arm[:k] == 1], y[:k][arm[:k] == 0]
        if min(a.size, b.size) < MIN_PER_ARM:
            path.append(np.nan)
            continue
        _, p = proportions_ztest([a.sum(), b.sum()],
                                 [a.size, b.size])
        path.append(p)
        final = p < ALPHA
        if p < ALPHA:
            any_hit = True
            favor_hit = favor_hit or a.mean() > b.mean()
    return final, any_hit, favor_hit, path if keep_path else None

g = np.random.default_rng(11)
res = np.array([one_trial(ctrl, g)[:3] for _ in range(TRIALS)])
print(f"{TRIALS} sham experiments, {LOOKS} looks each, "
      f"true effect exactly 0")
for i, label in enumerate(("FINAL look only          ",
                           "ANY look, either direction",
                           "ANY look, treatment ahead ")):
    print(f"  significant at {label}: {res[:, i].mean():.1%}")
print(f"  1 - 0.95**{LOOKS} (independent-look bound): "
      f"{1 - 0.95 ** LOOKS:.1%}")
```

Expected output: a header line stating the number of sham experiments, the number of looks, and that the true effect is exactly zero, then the three false-positive rates under the three stopping policies and the independent-look upper bound for comparison.

The function shuffles the control arm into a random arrival order, splits it at random into two sham arms, and walks forward through the data computing the same two-proportion test at fourteen evenly spaced moments — the analyst checking her dashboard each morning. Three counters record three different analysts. The disciplined one looks once at the end. The second stops at any significant difference in either direction. The third, who is the realistic one, stops only when the result is significant *and* the treatment arm is ahead, which is what actually happens in organizations because nobody halts a test to announce that the new idea is losing. Separating the second from the third is the correction this chapter's draft required: the simulation used to count either-direction hits while the prose described a treatment-is-winning policy, and those are different numbers answering different questions. The optional path return, which Code 11.11 switches on, then makes the phenomenon watchable rather than merely tabulated.

Code 11.11. Trace one analyst's p-value path

```
# The aggregate rates say how often peeking misleads. This
# says what it LOOKS like from the inside: one sham
# experiment, examined fourteen times, printed look by look.
g_one = np.random.default_rng(7)
final, any_hit, favor_hit, path = one_trial(
    ctrl, g_one, keep_path=True)

print("look   n per arm (approx)   p-value   would stop?")
for i, p in enumerate(path, start=1):
    approx = int(ctrl.size * i / LOOKS / 2)
    if np.isnan(p):
        print(f"{i:>4}   {approx:>16}   {'--':>7}   too small")
    else:
        stop = "YES <--" if p < ALPHA else ""
        print(f"{i:>4}   {approx:>16}   {p:>7.3f}   {stop}")
```

```
print(f"\ndisciplined analyst (final look only): "
      f"{'declares a difference' if final else 'declares nothing'}")
print(f"undisciplined analyst (any look)      : "
      f"{'declares a difference' if any_hit else 'declares nothing'}")
```

Expected output: a fourteen-row table giving the look number, the approximate customers per sham arm at that look, the p-value, and a flag marking any look at which an undisciplined analyst would have stopped; then one line each for what the disciplined and the undisciplined analyst would have concluded from the same experiment.

Input is the same function, called once with its optional path return switched on. The transformation is nothing; the output is the exhibit. Read the p-value column downward and watch it wander: on some seeds it dips below 0.05 in the middle of the window and climbs back out, which is the entire phenomenon in one column of numbers. The last two lines are the cell's argument. Two analysts, one experiment, identical data, opposite conclusions — and the only difference between them is a rule written down before launch. Run the cell across several seeds and note how often the two lines disagree.

VERIFICATION CHECK

Before you run: predict all three rates. The final-look rate should land near 5 percent, because that is what the threshold means and the data contain no effect — this number validates the simulation itself, so if it is far from 5 the rest is not interpretable. Then predict the any-look rate. The arithmetic of fourteen independent tests would suggest 1 − 0.95 to the fourteenth, about 51 percent, but the looks are not independent because each contains the previous one, so the true inflation is smaller. Predict where it lands, and then predict the favorable-only rate relative to it — the answer is close to half, and it is worth reasoning out why before you see it.

After you run: record all three. On this design the final-look rate sits near 5 percent, the any-direction rate near 22 percent, and the treatment-ahead rate near 11 percent; across seeds the three land in roughly the 4-to-6, 20-to-24, and 10-to-12 percent bands, so quote a band rather than a decimal. Then read the p-value path printed by Code 11.11 and describe what an analyst would have seen: a number wandering, dipping below 0.05 somewhere in the middle on some seeds, and climbing back out. That wandering is the entire phenomenon. Now write the three sentences the exhibit has earned. First, in a test with no effect whatsoever, roughly one analyst in nine who checks daily and stops when her treatment is ahead will ship a winner that does not exist. Second, that winner's report will look completely normal — a p-value below 0.05, an interval excluding zero, a plausible lift — because nothing in the final output records how many times the data were examined. And third, the defense is not statistical sophistication but the stopping rule in Table 11.3, declared before launch and honored afterward, which costs nothing and is skipped constantly.

Investigate if: the final-look rate is far from 5 percent, or the treatment-ahead rate is not roughly half the any-direction rate. The first means the simulation or the test call is wrong and no other number in the cell can be trusted. The second would be genuinely strange under a true null, because a symmetric null should produce significant results in both directions about equally often — so check that the arm comparison and the mean comparison use the same arm as arm one.

11.12.6 Lab 11.2, Part D: Practical Significance, Temporal Stability, and the Rollout Decision

The estimate is credible and both intervals are honest. This part converts them into the decision, stress-tests the economics that carry the conversion, and then applies the one durability diagnostic the certified file can actually support.

Code 11.12. Price the interval against break-even

```
# The DOSE comes first: the break-even line is undefined
# without it. Table 11.3 declares one alert per treated
# customer, so cost per assigned customer is one exposure.
COST_PER_ALERT   = 0.12
ALERTS_PER_CUST  = 1
CONTRIB_PER_CONV = 28.00

cost_per_cust = COST_PER_ALERT * ALERTS_PER_CUST
break_even = cost_per_cust / CONTRIB_PER_CONV
print(f"dose: {ALERTS_PER_CUST} alert(s) -> cost per assigned "
      f"customer ${cost_per_cust:.2f}")
print(f"break-even lift: {break_even * 100:.2f} pp")

print("\nvalue per 1,000 customers enrolled, at this dose")
for label, lift in (("interval low", d_lo),
                    ("point estimate", abs_lift),
                    ("interval high", d_hi)):
    incr = lift * 1000
    net = incr * CONTRIB_PER_CONV - 1000 * cost_per_cust
    print(f"  {label:<14s} lift {lift * 100:5.2f} pp"
          f"  {incr:5.1f} conversions  net ${net:8.2f}")
print(f"\ndecision rule met (lower bound > break-even): "
      f"{d_lo > break_even}")

# The line is an ASSUMPTION, so vary it. Contribution is an
# average over observed orders; incremental converters need
# not match it, and the rollout dose may differ.
print("\nbreak-even lift (pp) across dose and contribution")
print(f"{'contribution':>13}", end="")
for lab in ("1 alert", "2 alerts"):
    print(lab.rjust(9), end="")
print()
for contrib in (20.00, 24.00, 28.00, 32.00):
    print(f"{'$' + format(contrib, '.2f'):>13}", end="")
    for k in (1, 2):
        be = COST_PER_ALERT * k / contrib
        mark = "" if d_lo > be else "  <-- inside CI"
        print(f"{be * 100:>9.2f}{mark}", end="")
    print()
```

Expected output: the declared dose with the implied cost per assigned customer and the break-even lift; the incremental conversions and net contribution per thousand customers at the interval's lower bound, the point estimate, and the upper bound; whether the registered decision rule is met; and a sensitivity grid of break-even lifts across four contribution figures and two doses, flagging any cell the confidence interval does not clear.

Input is the interval from Code 11.9 plus three business constants; the transformation is the arithmetic of Section 11.7; the output is the decision rule declared in Table 11.3, evaluated, plus the sensitivity that keeps the evaluation honest. Note that the constants are inputs to the analysis and belong in the registration, not numbers chosen after the lift was known — an analyst who selects the contribution figure after seeing the estimate has invented a new way to peek. The sensitivity grid is the cell's real contribution to the deliverable, because it answers the question a finance partner asks second: not "does it pay?" but "how wrong would our cost and margin assumptions have to be before it stops paying?" On these numbers the answer is reassuring at one alert and much less so at two, and the flagged cells are where the conversation goes.

Code 11.13. Assess temporal stability

```
# NOT a novelty test. Customers assigned in the first and
# second halves of the window are different cohorts, exposed
```

```
# to different drops under different calendar conditions, so a
# gap between halves is not evidence of decay with exposure.
# Novelty needs repeated exposures for the SAME assignment,
# which this one-outcome-per-customer file does not contain.
span = ab["assign_date"].max() - ab["assign_date"].min()
half = ab["assign_date"].min() + span / 2
ab["cohort"] = np.where(ab["assign_date"] <= half,
                        "first half", "second half")

for cohort, grp in ab.groupby("cohort"):
    gs = (grp.groupby("variant")["converted"]
             .agg(n="size", x="sum"))
    r = gs["x"] / gs["n"]
    lift = (r["treatment"] - r["control"]) * 100
    per_arm = int(gs["n"].min())
    se = np.sqrt(sum(p * (1 - p) / n for p, n
                     in zip(r, gs["n"]))) * 100
    print(f"{cohort:<12s} n/arm {per_arm:>5d}"
          f"  control {r['control']:.2%}"
          f"  treatment {r['treatment']:.2%}"
          f"  lift {lift:+.2f} pp  (SE {se:.2f})")

print("\nRead the two SEs before reading the two lifts: at "
      "roughly half\nthe arm size, each half-window estimate "
      "is far too imprecise to\ndetect a 1.8-point effect, so "
      "the halves WILL differ by luck.\nThis diagnostic can "
      "flag a difference big enough to investigate.\nIt cannot "
      "certify stability, and it cannot measure novelty.")
```

Expected output: one line per assignment half giving the per-arm sample size, the control and treatment rates, the estimated lift in percentage points and its standard error; then the standing note on what the diagnostic can and cannot establish.

The cell splits the experiment by when customers were assigned and re-estimates the lift in each half. Three things about it are corrections to the draft of this chapter. It is named a temporal-stability diagnostic rather than a novelty check, because comparing cohorts assigned on different dates cannot isolate a response to unfamiliarity. It prints each half's standard error beside its lift, so the reader sees the imprecision before the difference rather than after. And its closing note states plainly that the file's single binary outcome per customer contains no repeated-exposure information at all, which is why the rollout plan must commission the multi-exposure design Section 11.8 describes rather than treating this exhibit as a substitute for it.

VERIFICATION CHECK

Before you run: predict the two half-window lifts and, more importantly, predict their standard errors. With roughly 2,500 customers per arm per half, the standard error of each half's difference is close to 0.72 percentage points, which means the two halves can differ by a point and a half without anything having happened. Write down, before you look, what difference between halves would actually warrant investigation given that imprecision.

After you run: compare the two lifts against the two standard errors, and write the interpretation in one sentence: the half-window estimates are too imprecise to detect novelty decay, so this diagnostic can only flag a difference dramatic enough to demand investigation and cannot certify its absence. Then state what the rollout plan must therefore carry — a scheduled re-measurement, and a design that follows one randomized assignment across several consecutive drops so that a treatment-by-exposure pattern is estimable at all. Finally, note what this whole part has done to the CMO's proposal: it supports expanding the Drop Alert at the tested dose, and says nothing whatever about defunding paid social, per Section 11.7's third restriction.

Investigate if: the two halves differ by more than about three standard errors, or either half's control rate is far from 6 percent. The first is large enough to be worth tracing to a calendar event, a promotion, or a delivery incident in one half of the window. The second suggests the assignment window overlapped something — a holiday, a site outage, a competing campaign — that the registration did not anticipate, which belongs in the deliverable's deviations list.

11.12.7 Lab 11.2, Part E: The Round Trip and the Rollout Paragraph

The last part is the assistant round trip of Section 11.11, run end to end, and it is where the chapter's deliverable is actually written. Work in four moves, documenting each per Appendix D.

First, the registration prompt. Draft the preregistration for a *second* experiment — the one Section 11.7's third restriction says StyleCraft needs, a paid social incrementality test with a geographic holdout — using the template of Table 11.3 and the instruction that the assistant must write UNSPECIFIED and ask rather than filling gaps. Count the UNSPECIFIED rows it returns, and check specifically whether it left the dose and the guardrail-computability rows unspecified or invented them. That count is a measurement of how much of your own design was missing from your prompt, and it is the most useful number the assistant will produce all week.

Second, the corrupted-analysis audit. Hand the assistant the ab_test file with a deliberately misleading prompt: ask it to "check whether the campaign worked, and see which customer segments responded best." Then audit what comes back against Table 11.7. Expect four specific failures, and record whether each occurred: a segment harvest with unadjusted p-values across home_metro, generation, and loyalty_tier; a headline built on whichever subgroup won; no mention anywhere that the subgroups were not registered; and no statement of the treatment dose the reported lift belongs to. The exercise is not a trick. It is a demonstration that the assistant answers the question it was asked, and that the analyst's job is to ask the registered question.

Third, the adversarial prompt. Ask the assistant to argue against your own result: every rival explanation, every contamination path, every difference between the tested population and the rollout population, and every way a rollout at two alerts per week differs from a test at one. Keep the items that survive your judgment and place them in the deliverable's limitations section, cited to nothing — they are your limitations now.

Fourth, the drafting prompt, with only the audited numbers pasted in.

VERIFICATION CHECK

Before you run: write down what you expect the assistant to get wrong, item by item, against Table 11.7's three supplemental points. Predicting the failure modes before you see them is what turns the round trip from a demonstration into an audit, and it is the part students most often skip.

After you run: run every drafted paragraph against three tests. The verb test, per Section 11.10: the paragraph may say the Drop Alert *caused* a lift among the tested population at the tested dose within the stated interval; it may not say the rollout *will* lift conversion by 30 percent. The number test: every figure in the paragraph must appear in a lab output above, with the absolute lift stated before the relative one, the control rate beside it, and the relative interval computed as a ratio rather than rescaled. The qualification test: the paragraph must name the tested population, the tested window, and the tested dose, because dropping any of the three converts a defensible finding into the deck that opened this chapter. Then write the final version yourself. The deliverable's key sentence, in the pattern Section 11.13 will polish: among SMS-opted-in customers, over a fourteen-day window, at one Drop Alert per customer, the alert raised conversion from 6.0 percent to 7.8 percent — a lift of 1.8 percentage points, 95 percent interval 0.8 to 2.8 points, or roughly $106 to $662 of contribution per thousand customers per drop window against a break-even of 0.43 points — and the program therefore pays for itself across the full range of effects the experiment supports, subject to the stated cost and contribution assumptions, the extrapolation limits, and the re-measurement schedule below.

> Investigate if: the assistant's draft is clean on the first pass. That is possible and it is not a reason to relax, because the failure modes of Section 11.11 are silent rather than loud. Check the two that hide best: whether any reported interval is a rescaled difference interval wearing a ratio's label, and whether the sample size implied by the interval's width matches the number of randomized customers rather than the number of orders.

11.13 Marketing Interpretation and Managerial Insight

The labs produced a certificate, two intervals, and a break-even comparison; the reallocation meeting runs on sentences, and the sentences an experiment produces are unusually easy to say wrong. This section translates, and it does so partly by exhibiting the wrong readings and correcting them, because experimental results attract a particular family of misinterpretation: they arrive with the authority of a trial, which makes stakeholders more rather than less willing to stretch them past what was tested.

The first wrong reading is the one this chapter's outline predicted, and it will be spoken within a minute of the result going up: "It's significant at p less than point-oh-five, so the campaign will lift sales 12 percent at scale." Five errors are packed into a single clause, and each has a section behind it. The p-value has been read as a statement about the future and about magnitude, when it is neither — it is a measure of how surprising this difference would be if the true effect were zero, and Section 7.10's warnings apply unchanged (Wasserstein & Lazar, 2016). The 12 percent has arrived from nowhere; the experiment measured a 30 percent relative lift on conversion, with an interval running from roughly 12 to 50 percent, and conversion is not sales — revenue per assigned customer was a declared *secondary* metric precisely because a treatment can raise conversion while lowering average order value, and the deliverable reports it as exploratory rather than promoting it to the headline. The words "at scale" quietly change the population from SMS-opted-in customers with thirty-day accounts to whatever the rollout reaches, which the experiment did not test. The claim carries no dose, so it silently transports a one-alert result onto a rollout the CRM team wants to run at two, which is a different intervention with twice the cost. And "will" has replaced "caused," which converts a bounded past-tense measurement into a promise, exactly the move Section 10.15 called laundering when a forecast's interval was stripped in the retelling. The corrected sentence is longer and worth the length: among SMS-opted-in customers, over a fourteen-day window, at one Drop Alert per customer, the alert caused conversion to rise 1.8 percentage points from a 6.0 percent base — with the evidence consistent with anything from 0.8 to 2.8 points — and a chain-wide quarter-long rollout at higher frequency would be expected to deliver somewhat less per alert, for reasons the limitations section names.

The second wrong reading is quieter and destroys more experiments than the first, because it ends programs instead of overselling them: "The test wasn't significant, so the campaign doesn't work." An experiment that fails to reject is not an experiment that proves the null, and the distinction is entirely a matter of power. Code 11.7 makes the point in numbers rather than in principle: a test with 1,000 customers per arm, run on this same genuinely valuable 1.8-point treatment, would have found it only about a third of the time, and a test at 500 per arm could only have reliably detected a lift of nearly 5 points. Reporting either outcome as "no effect" retires a profitable program on the strength of an experiment that was never capable of finding it. The standing rule this guide adopts: a null result is reported with its minimum detectable effect attached. "The test found no significant difference; it was powered to detect a lift of 3 points or more, so effects smaller than that remain entirely possible" is an honest sentence, and it is a different recommendation — usually "test again, larger" rather than "stop."

The third wrong reading belongs to the subgroups, and it is the most seductive because it feels like insight: "The overall lift was modest, but look — it was 6.2 points among Gen Z suburban app users. Let's target them." Section 11.8 named the mechanism, and the deliverable's defense is structural rather than rhetorical. Subgroup results appear in an exploratory section, visually separated, with a standing note that they were not registered, that many were examined, and that the largest of many estimates is biased upward by the act of being the largest. What they license is a next experiment, registered in advance on that subgroup, which is a cheap and excellent use of the finding. What they do not license is a targeting decision, and an analyst who lets an unregistered subgroup drive a rollout has spent the credibility the whole design was built to earn.

The deliverable that survives all three misreadings has a fixed anatomy, assembled entirely from lab outputs, and its ordering is itself an argument. Page one is the registration, dated, with the treatment dose in the same table as the primary metric and any deviations from the plan listed as deviations — not because deviations are disqualifying but because undisclosed ones are. Page two is the assignment certificate from Lab 11.1 Part B: the assignment-flow record from randomization through outcome observation, realized arm sizes against the declared allocation, the covariate balance profile read as a diagnostic, the sham-split analysis-path check with a note that a genuine end-to-end A/A test remains outstanding, and the contamination paths named with their expected direction. Only then, page three, the primary result: control rate, treatment rate, absolute lift with its interval on the difference, relative lift second with its own interval on the ratio, and the same difference interval restated in incremental conversions and dollars per thousand customers at the tested dose. Page four is the decision: the break-even lift computed from the program's own dose and economics, the comparison of the full interval against that line, the sensitivity of that line to contribution and dose, and the recommendation stated as a rule rather than a hope — expand at the tested dose, because the program pays across the entire interval under the stated assumptions, with the asymmetric-cost reasoning of Section 2.7 made explicit. Page five is the limitations, which for an experiment are unusually specific and unusually important: the tested population against the rollout population, the fourteen-day window against a quarter, the one-alert dose against the proposed frequency, the novelty risk with the multi-exposure design that would actually measure it, and the sentence the CMO will not enjoy — that the reallocation from paid social requires an incrementality estimate for paid social that does not exist, and that the correct next step is the experiment which would produce it. Page six is the guardrails and the reversal plan: the revenue-per-assigned-customer threshold declared in Table 11.3, who monitors it, what happens if it trips, and the instrumentation obligation that would let the next test declare an unsubscribe guardrail it can actually compute. And the exploratory section, last and labeled, holding the subgroups and secondary metrics as candidates for the next registration.

One further translation deserves its own paragraph, because it is what makes this chapter's deliverable different in kind from the four before it. Chapters 8, 9, and 10 produced artifacts whose value was the number inside them. This chapter's artifact is valuable mostly for a decision it makes possible in the *other* direction: the cheapest thing an experiment does is kill a bad idea for the price of a test rather than the price of a quarter. StyleCraft's Drop Alert survived. Had the experiment returned an interval from −0.4 to +0.9 points, straddling the break-even line, the deliverable would have saved a four-hundred-thousand-dollar reallocation for the cost of ten thousand messages, and that is the outcome an experimentation practice earns its budget on. Managers judge experiments by the winners they find. Analysts should judge them by the losers they catch, and should say so in the meeting, because it is the argument that funds the second experiment.

11.14 Business Analytics in Practice

This section turns from the fictional case to how controlled experimentation operates — and fails — in professional organizations. It looks at where experiments have become the default evidence standard at the largest technology companies, where their absence has quietly distorted a decade of advertising budgets, and where the discipline that looks pedantic in a textbook turns out to be the entire difference between a testing program and a rumor mill. The first two vignettes draw on published research and documented industry practice; the third is a composite of recurring commercial testing patterns rather than a report about a single named organization.

11.14.1 Experimentation Culture as Infrastructure

The first vignette is the experimentation culture, which at the companies that pioneered it is now a piece of infrastructure rather than a practice. Microsoft's Bing organization, Booking.com, Amazon, and their peers run controlled experiments continuously and at a scale that is difficult to picture from a marketing department: thousands of concurrent tests, a substantial fraction of all product and messaging changes shipped only after passing one, and platforms that handle assignment, delivery, and analysis so that the analyst's judgment is spent on design rather than plumbing (Kohavi et al., 2013; Kohavi et al., 2020).

Three of the practices these organizations converged on are directly borrowable by a team of four with no platform at all. The first is a single declared overall evaluation criterion — one metric that decides, chosen in

advance, resisting the pull toward a dashboard of twelve — which is the preregistration's primary-metric row operating as culture rather than paperwork. The second is guardrail metrics: measures that are not expected to improve but must not degrade, such as unsubscribe rate, page latency, or margin, checked on every test so that a conversion win bought at the cost of a churn increase is caught at the test rather than at the quarterly review. The corollary that matters for a small team is instrumentation: a guardrail is only a guardrail if the pipeline delivers the field, which is why mature programs treat "we could not compute that" as a defect with an owner rather than a footnote. The third, and the most transferable, is the standing A/A test: the same organizations that run thousands of experiments run continuous null experiments through the full production pipeline, because a randomizer that silently breaks invalidates everything downstream of it and the only way to know is to keep asking a question whose right answer is already known. Note that this is the end-to-end version, not the analysis-path check a spreadsheet can do — the point is that the assignment, the send, and the tracking are all in the loop. The practice lesson for a small team is proportion, not scale. Nobody needs a platform to declare one metric, name a guardrail the pipeline can actually measure, and split a list in half through the real send machinery twice a year to check that the splitter still works.

11.14.2 Advertising Incrementality and the Measured-Channel Bias

The second vignette is advertising incrementality, and it is the one with the largest number attached. The industry standard for evaluating digital advertising has long been attribution: assigning credit for conversions to the ads that preceded them, using data the platforms themselves supply. Attribution is observational, and Section 11.3 explains exactly what is wrong with it — ads are shown to people the targeting system predicts are most likely to convert, so many of the conversions that follow would have happened anyway, and the resulting measurement credits the advertiser for its own customers' intentions. This is the upward-pointing case of selection bias, and it is upward for a specific and identifiable reason: the selection mechanism targets high-propensity customers.

When experiments have been run against that standard, the gap has been large. eBay's large-scale field experiments on branded paid search found that returns were far lower than the attributed figures suggested, with much of the measured value reflecting traffic that would have arrived through other channels (Blake et al., 2015). Lewis and Rao (2015) established the structural reason the industry rarely notices: sales outcomes are so variable relative to advertising's true effect that credibly measuring the return requires experiments far larger than firms typically run, which means most advertising ROI numbers in circulation come from tests without the power to produce them. Gordon et al. (2019) closed the loop by running experiments and observational analyses on the same campaigns with rich covariates available, and found the observational estimates diverged from the experimental ones by margins large enough to reverse budget decisions — and, importantly for Section 11.3's argument about direction, diverged in both directions rather than consistently upward. The industry's response is the geo experiment and the holdout: randomizing markets rather than individuals when the medium cannot be targeted, or withholding advertising from a randomly chosen share of the audience and comparing (Vaver & Koehler, 2011).

The practice lesson lands on Section 11.7's third restriction, and it is the most valuable sentence a junior analyst can say in a budget meeting. A marketing organization that has measured one channel experimentally and the rest by attribution will systematically move money toward the measured channel — not because it performs better but because it is the only one whose number was ever tested. The bias is not in any single estimate. It is in the comparison, and it favors whichever program was brave enough to be measured.

11.14.3 The Testing Program That Manufactured Knowledge

The third vignette is smaller, is included because most students will encounter its version before they encounter either of the others, and is presented as a composite vignette built from recurring commercial testing practices rather than as a report about a specific firm. A mid-size retailer's CRM team runs subject-line tests on its weekly newsletter, checking results on a dashboard each morning and declaring a winner as soon as one appears, which typically happens within a day or two. Over a year the program accumulates an impressive log of victories — dozens of winning lines, each with a lift of eight to fifteen percent — and the aggregate newsletter conversion rate does not move at all. When the team eventually re-runs a sample of its past winners as fresh, properly sized, single-look tests, most of them reproduce nothing.

The mechanism is Section 11.8's first failure, operating exactly as Code 11.10 stages it: with a small daily audience and a stopping rule of "stop when it wins," the program is not testing subject lines but sampling noise until the noise is flattering, and the asymmetry of only ever stopping on wins converts an unbiased random process into a machine that produces a winner far more often than the nominal threshold suggests (Johari et al., 2017). The costly part is not the wasted year. It is that the team has built a set of beliefs about its own customers — what tone works, what urgency works, what emoji works — out of pure noise, and those beliefs shape creative decisions well beyond the newsletter. The practice lesson: a testing program without a stopping rule does not merely fail to learn, it actively manufactures false knowledge, and it does so faster the more tests it runs.

11.14.4 In Your First Analyst Job

In your first analyst job, these vignettes compress into one expectation, and it is this chapter's closing thread: the cheapest experiment stops a bad idea before rollout. The glamorous version of experimentation is the winner — the change that lifted conversion, the test that justified the budget, the slide with the green arrow. The version that pays for the practice is the other one: the redesign that tested flat and did not ship, the offer that tested negative and was withdrawn, the reallocation that a ten-thousand-customer test showed was resting on a number three times too large. Those results generate no celebration and no slide, and they are where nearly all of the return lives, because the cost of testing an idea is always a small fraction of the cost of rolling one out. The analyst who understands this argues for experiments differently from the one who does not. She does not promise that the test will find a winner, which she cannot know. She promises that the organization will find out cheaply either way — and that promise, unlike the other one, is one she can keep every time.

11.15 Ethics, Testing on Customers, and the Fairness of the Holdout

The Business Analytics in Practice section ended on experiments as a way to fail cheaply; this section examines what it means that the failing is done to people who did not volunteer. It extends the guide's running discussions — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), causal language (Section 7.15), acting on predictions about people (Section 8.15), fairness across groups (Section 9.15), and forecast accountability (Section 10.15) — to what experimentation adds, which is a category the previous chapters did not contain. Every prior chapter analyzed customers. This chapter's methods do something to them, deliberately, by a chance mechanism, and specifically in order to see what happens.

Start with the fact that makes the ethics unavoidable rather than optional, and state it precisely, because the loose version of the sentence gets the ethics backward. An experiment knowingly assigns customers to different experiences while the organization remains genuinely uncertain which is better overall. It does not assign anyone to a worse experience on purpose — if the company already knew which arm was better, there would be nothing to learn and no ethical basis for withholding it. Some customers will retrospectively turn out to have received the inferior option, and that is a real cost, but it is not knowable at the moment of assignment, and the uncertainty is precisely what makes the test defensible rather than what makes it suspect. The control arm in StyleCraft's test receives the existing standard drop announcement — the experience every customer received last month and most customers still receive — not a degraded one. And the customer in the control arm was not asked. She agreed, somewhere in a terms-of-service document she did not read, to receive marketing communications; she did not agree, in any sense the word normally carries, to be a comparison group.

The reference case for what happens when this is taken lightly is the emotional contagion study, in which the news feeds of hundreds of thousands of users were altered to show more or less emotionally positive content and the effect on their own posting was measured (Kramer et al., 2014). The research was competent, the effect was real, the manipulation was small, and the publication triggered a public and academic reaction severe enough that the journal issued an editorial expression of concern about the consent procedures (Verma, 2014). The instructive part for a marketing analyst is not the scientific ethics debate but the commercial one. What the public objected to was not the size of the effect; it was the discovery that the emotional content of their feed was something a company would deliberately vary to see what happened to them. The line the episode drew is not about statistical harm. It is about

the difference between a company improving its product and a company running an experiment on its users' emotional states, and that difference is legible to ordinary people even when it is invisible in the analysis plan.

This suggests the working distinction the guide adopts, which is coarse but usable. Testing which of two versions of a product experience serves customers better is ordinary business improvement, the kind every company has always done informally and which a controlled experiment merely does honestly. Testing something whose mechanism is the customer's psychology rather than the product's quality — manipulating scarcity signals, urgency, emotional framing, or the perception of price — is a different activity wearing the same clothes, and it deserves a harder question before it runs: would we be comfortable explaining this test, in plain language, to the customers in it? The question is not a bright line and it is not meant to be. It is a prompt that reliably separates the tests nobody will mind from the tests that end up in a journalist's inbox, and an analyst who asks it out loud in the design meeting has done the most useful thing available.

Marketing's own hardest case is not emotional framing but price and offer testing, and it deserves naming because students will meet it. Randomizing which customers see which discount is a real experiment with real value, and it also means charging different people different amounts for the same item on the basis of a chance draw. The practices that keep this defensible are specific. Test discount *depth* on offers rather than base prices, so nobody pays more than the posted price. Keep the treatment duration short and the arms' economic difference modest. Honor the losing arm afterward where the difference was material, which is the practice of extending the winning offer to the control group once the test closes; that practice is not required by any statute and is the clearest available signal that an organization understands what it just did. And handle protected characteristics with the precision the fairness discipline of Section 9.15 established, which the draft of this chapter did not. The rule is not that the assignment variable must never correlate with a protected characteristic — a valid chance mechanism will show small realized correlations with everything, by luck, and demanding zero correlation would forbid randomization itself. The rule is that the assignment mechanism must not use protected or proxy characteristics as inputs, and that the realized distribution of benefits, burdens, offers, and error rates should be audited across relevant groups after the fact, with a stated response if the audit finds a pattern.

Which brings the section to the holdout, and to the question the chapter poses most sharply: who is denied the benefit, and why is that defensible?

CONCEPT

The Holdout Is a Debt You Repay

A control arm is customers who did not receive something the company hopes is good, held there so the company can find out whether it actually is. The arrangement is defensible under four conditions, and the analyst should be able to state all four before the test launches. First, genuine uncertainty: the treatment's value is unknown, which is precisely why the test is being run — withholding a benefit already known to be beneficial is not an experiment but a cost saving with a scientific alibi. Second, proportion: the holdout is no larger and no longer than the power calculation requires, which is the ethical function of Code 11.7's arithmetic — an oversized or open-ended holdout denies more people more benefit than the question needs, and an undersized one denies them the benefit and learns nothing, which is worse.

Third, bounded harm: the control arm receives the existing experience rather than a degraded one, and any guardrail that would signal real damage is declared in advance, computable from the data the pipeline actually delivers, and attached to a rule for stopping. Fourth, repayment: when the treatment wins, the holdout receives it, promptly, and the plan to do so exists before the test starts rather than being remembered afterward. Under those four conditions the holdout is the cost of finding out, borne briefly and repaid — and an organization that cannot meet all four should notice that its discomfort is information about the test rather than squeamishness about statistics.

Source: Course concept developed for this guide, informed by Kohavi et al. (2020) and Kramer et al. (2014).

Two smaller obligations close the section, and both are about transparency in directions people forget. The first is internal. An experiment's result is a fact about the business, and the temptation to report only the tests that won is enormous — it is the peeking problem of Section 11.8 relocated from the analysis to the archive. An experimentation practice that keeps a complete log, including the flat results and the losses, is doing the same thing the override log of Section 10.9 did for forecasts: making its own judgment auditable over time, so the organization can learn which of its instincts deserve to keep being funded. The practice that files only winners is not merely dishonest; it is the mechanism by which the retailer in Section 11.14.3's composite vignette built a year of false beliefs.

The second obligation is external, and it is a governance recommendation rather than an empirical claim — a distinction the draft of this chapter blurred and this revision states plainly. It is not universally true that customers cannot be asked to consent to every A/B test, nor that almost no marketing organization discloses experimentation, nor that disclosure has been shown to cost nothing; those are three assertions of fact that would each need a source, and the sources do not exist in the form the claims require. What is true is that consent and disclosure obligations vary, and that the variation follows identifiable factors: the jurisdiction and its privacy regime, the risk the intervention carries, the sensitivity of the outcome being measured, whether the activity is framed as research or as ordinary product improvement, the representations the company has already made in its own privacy and contractual language, and whether the population includes people entitled to additional protection. Some tests in some jurisdictions require explicit consent; many ordinary product-improvement tests do not; and the boundary is a legal and ethical question that an analyst should route to someone qualified rather than resolve from a textbook. Against that background, this guide recommends the plain-language transparency practice as a matter of governance: state, in a privacy or research statement written in language a person would actually read, that the company tests variations of its experiences and messages and that customers may at times receive different versions. The recommendation converts a thing that looks bad when discovered into a thing that was disclosed all along, and an organization that finds itself unwilling to write that sentence has learned something about its own testing program.

11.16 Chapter Summary

This chapter closed Part II by building the only instrument in the guide that produces evidence about actions rather than patterns. It began from an impossibility: a customer's causal effect is the difference between her outcome under treatment and her outcome without it, exactly one of those outcomes ever occurs, and the other — the counterfactual — is not missing from the file but missing from the world. That fundamental problem redirected the enterprise from measuring effects within units to constructing comparisons across them, and the construction is where the chapter's real content lives.

Selection bias named what goes wrong when the comparison is inherited rather than constructed. The Drop Alert's spectacular 5.30-point observational difference decomposed, on four cells of StyleCraft's own record, into a 1.80-point adjusted difference and a 3.50-point composition component — the CRM team's targeting competence read back to it as campaign performance. Two qualifications travel with that decomposition and both are load-bearing. The adjusted 1.80 points is a causal effect only under the miniature's designed assumption that engagement is the sole confounder, measured without error, with no unmeasured selection remaining; it sits on the third rung of Section 11.10's ladder, not the fifth. And the direction of selection bias is not a constant. It points upward when high-propensity customers are preferentially treated, as here, and it points the other way for win-back, save, and onboarding programs that deliberately treat the customers least likely to convert.

Random assignment was then introduced as the device that removes the problem wholesale rather than variable by variable, balancing in expectation every characteristic a customer has, including the ones nobody measured or imagined, at the price of deciding by chance what a competent marketer would rather decide by judgment. The chapter was careful about what that purchase includes. Randomization delivers comparability; an unbiased effect estimate additionally requires that assignment was implemented as planned, that outcomes are observed consistently, that units are analyzed as assigned, that no unit's outcome depends on another's assignment, and that the treatment is well defined at a stated dose. StyleCraft's design was specified as complete randomization — exactly five thousand

of ten thousand eligible customers drawn for treatment — which is what makes a realized arm-size shortfall a finding rather than noise.

Around that single move the chapter assembled the protections without which it does not survive contact with an organization. The control arm was defined as the real alternative rather than silence. The unit of randomization was chosen large enough to contain the treatment's spillover, with the reminder that outcomes recorded at a finer grain than the assignment are more description rather than more randomized evidence. The preregistration fixed metric, arms, dose, mechanism, sample size, duration, decision rule, and above all stopping rule before any outcome was visible, as the experimental form of Section 2.5's analytic specification — and every guardrail it declared was one the certified data can actually compute, because a preregistered guardrail that cannot be calculated is a design defect rather than a missing nicety. The power calculation turned a minimum detectable effect into a sample size, and Code 11.7 made it executable: 4,379 customers per arm for the registered 1.5-point effect, 94.5 percent power against the planted 1.8 points at the registered 5,000, and 35.6 percent at a thousand — the number that turns "no significant difference" into a statement about the experiment. The intention-to-treat rule kept every randomized customer in her assigned arm, with an assignment-flow record reported from randomization through outcome observation, because dropping the customers whose messages failed is precisely how an analyst converts a randomized comparison back into an observational one. Covariate balance was reframed as a diagnostic rather than a test, since chance imbalance occurs under a valid procedure and a conspicuous imbalance calls for an audit of the assignment code rather than a re-randomization. And the lab's sham-split procedure was renamed for what it is — an analysis-path check on completed control-arm outcomes — with a genuine end-to-end A/A test named as the separate obligation it remains.

The measurement itself was almost anticlimactic: two proportions, an absolute difference of 1.80 percentage points, a 95 percent interval from 0.81 to 2.79, a relative lift of 30 percent whose own interval — computed as a ratio rather than by rescaling the difference interval — runs from roughly 12 to 50 percent. The chapter's verification theme paid off exactly there, with the estimate recovering the true lift the dataset was built to hide in plain sight. The interval was then converted into a decision by a line the data cannot supply, and the line required a fact the draft of this chapter had left unstated: the dose. At one alert per treated customer, twelve cents against a twenty-eight-dollar contribution gives a break-even lift of 0.43 points, the whole interval clears it, and the recommendation is robust rather than hopeful. At two alerts the line moves to 0.86 points, the interval's lower edge no longer clears it, and the same evidence supports a different recommendation — which is why the dose belongs beside the metric in the registration, and why the break-even comparison ships with a sensitivity grid across cost and contribution instead of a single number.

Five failure modes were cataloged as the standing ways an honest design is destroyed after the fact — peeking, multiple comparisons, mid-test changes, novelty, and contamination — and the labs made the first tangible by manufacturing false winners out of StyleCraft's control arm and nothing else, separating the roughly 22 percent chance of seeing a significant difference in either direction across fourteen looks from the roughly 11 percent chance of seeing one with the treatment ahead, which is the policy an actual analyst follows. Novelty was given its honest treatment: a comparison of customers assigned in the first and second halves of the window is a temporal-stability diagnostic, not a novelty test, because it compares different cohorts exposed to different drops, and the single binary outcome the certified file carries contains no repeated-exposure information at all. Measuring novelty requires following one randomized assignment across several exposures, which the rollout plan must commission.

A bounded section handled the decisions no chance mechanism can reach, with pre/post-plus-comparison and difference-in-differences taught as arithmetic and their parallel-trends assumption stated as the unverifiable claim it is. And the evidence hierarchy finally placed all of Part II on a ladder, not to rank the chapters' usefulness but to constrain their verbs: descriptions describe, models predict, quasi-experiments suggest under assumptions, and only experiments cause — within a tested population, a tested window, and a tested dose, every one of which the retelling will try to drop. The hierarchy carries one condition the chapter states rather than assumes: it describes the warrant available from competently executed designs, and a randomized experiment wrecked by attrition, noncompliance, interference, outcome switching, or repeated peeking can deliver less than a careful quasi-experiment. The rung is set by the design. The warrant is set by the design and the execution together.

Looking ahead, Part II is now complete, and its output is a shelf of evidence: a segmentation, a driver model, a scored list, a churn program with a threshold, a forecast with an interval, and an experimentally measured lift with a rollout rule. Every one of those artifacts is currently a table, a metric, or a paragraph, and every one of them has to survive a room. That is the problem Part III takes up. The next chapter turns to data visualization for analysis, where the first job of a chart is not to persuade anyone of anything but to let the analyst see what the numbers contain — and where the guide's verification discipline meets a new adversary, the AI-drafted chart that is fluent, attractive, and subtly wrong about the very expansion story these chapters have spent eleven chapters establishing. Between here and there sits Project #1, in which the groups assemble an end-to-end analysis from Part II's machinery and defend it out loud; the exercises below are written with that deliverable in view.

11.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: ask how each side of a comparison came to exist before quoting the comparison; declare the metric, the dose, and the stopping rule before the data arrive; size the test against the smallest effect worth finding; audit the assignment and keep every randomized customer in her assigned arm; report absolute lift with its baseline and its interval, and give a relative figure an interval of its own; compare the whole interval against a break-even line computed in advance from a stated dose; and keep the verbs on the rung the design earned. They are organized into three groups. Core chapter practice is the required path and should be completed by every student, and it holds the two homework submissions from which your instructor will assign a subset; Self-Assessment #6 draws on the concept and interpretation material of this chapter. In-class activities are prepared for discussion rather than submitted. Extensions are optional. Each exercise carries its assignment label so that instructors can assign selectively. Note that several of these exercises feed Project #1 directly — groups pursuing the causal track will submit a version of Exercise 11.5 as part of their deliverable, and the rubric in Appendix F applies.

11.17.1 Core Chapter Practice

Exercise 11.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- State the fundamental problem of causal inference, and explain why it is not solved by collecting more data about the same customers.
- Define selection bias, explain why competent marketing targeting reliably produces it, and name two marketing programs whose targeting rule would bias an observational estimate downward rather than upward.
- Explain what random assignment guarantees about characteristics the company has never measured, and then name three of the five additional conditions an unbiased effect estimate requires beyond comparability.
- Distinguish Bernoulli randomization from complete randomization, and explain why the distinction determines whether an arm-size shortfall is a finding or a coincidence.
- State the intention-to-treat principle. Then explain, in one sentence, why dropping the customers whose messages failed to deliver converts a randomized comparison into an observational one.
- A colleague sees a 1.4 percent difference in mean prior spend between arms and proposes to re-randomize. Explain why this is a mistake, what the imbalance does and does not tell her, and what the two legitimate responses are.
- Explain why a sham split of completed control-arm outcomes is not an A/A test, and list three parts of the apparatus it does not grade.
- A team randomizes by session but reports conversion per customer. Name the failure and state what it does to the interval's width.
- Distinguish absolute from relative lift, explain which one scales to business volume, and state the assumption under which a relative lift can be carried to a population with a different baseline rate.

- Explain why a confidence interval for a relative lift may not be obtained by dividing the endpoints of a difference interval by the observed control rate.
- Explain the difference between statistical and practical significance, and state the two quantities that must both be declared before a break-even lift can be computed at all.
- Explain why an analyst who checks results daily and stops on significance has a false-positive rate far above 5 percent, and explain why the rate for an analyst who stops only when her treatment is ahead is roughly half the rate for one who stops on any significant difference.
- A test returns no significant difference. Write the honest one-sentence report, including the element Section 11.13 requires that most reports omit.
- Explain why comparing the lift among customers assigned in the first half of an experiment's window with the lift among those assigned in the second half does not test for a novelty effect, and describe the design that would.
- State the parallel trends assumption of difference-in-differences, and explain why showing several parallel pre-periods supports it but does not prove it.
- Place these four claims on the evidence hierarchy and rewrite any whose verb outruns its rung: "VIP members spend 40 percent more"; "the model predicts churn with an AUC of 0.78"; "the store opening caused a $5 per-customer lift relative to comparison markets"; "the early-access alert lifted conversion 1.8 points."
- The evidence hierarchy puts randomized experiments above quasi-experiments. Describe a randomized experiment whose evidence you would trust less than a well-executed difference-in-differences, and name the execution failures that would put it there.

Exercise 11.2 The Observational Trap by Hand (Required Practice)

Using only the four-cell record of Table 11.1 and no code, rebuild every figure in Code 11.2's Verification Check: the four stratum rates, the two within-stratum differences, the two aggregate rates, the naive 5.30-point difference, the constructed counterfactual of 8.25 percent, and the split of 5.30 into a 1.80-point adjusted difference and a 3.50-point composition component. Then show the composition component a second way from mix alone, using the 50-point difference in engagement mix and the 7.00-point untreated gap between engaged and unengaged customers.

Next, change one number and re-derive. Suppose the CRM team had sent the Drop Alert to 5,000 engaged and 3,000 unengaged customers instead of 6,000 and 2,000, with all four stratum rates unchanged. Recompute the treated aggregate rate, the naive difference, and the composition component, and state in one sentence what your result shows about the relationship between targeting precision and the size of the bias.

Then answer the question the chapter cares about most. You have just recovered a 1.80-point adjusted difference from observational data, and the experiment will later recover a 1.80-point effect. Write two sentences explaining why these are not the same finding, naming the assumptions the first one rests on and the rung each occupies on Table 11.6. Finally, write the two-sentence memo you would send the finance partner, using no statistical vocabulary at all.

Exercise 11.3 Lift, Intervals, and the Break-Even Line (Required Practice)

A different StyleCraft program — a post-purchase SMS asking for a review, hypothesized to raise repeat purchase — was tested on 4,000 customers per arm at a dose of one message per assigned customer. Control: 240 repeat purchasers. Treatment: 296 repeat purchasers. The message costs $0.09 per customer and StyleCraft's average order contributes $28.

Compute the control and treatment rates, the absolute lift in percentage points, and the relative lift. Compute the standard error of the difference and the approximate 95 percent confidence interval on the difference; two standard errors either side is sufficient at this level. Then compute an interval for the relative lift the correct way, as a ratio, using the log construction of Code 11.9, and compare it against the interval the rescaling shortcut would have produced — report both and state which one the deliverable carries. Compute the break-even lift at the stated dose and state whether the decision rule of Table 11.3 is met. Then write the recommendation sentence, including the dose — and note before you start that the pattern from Lab 11.2 Part E may not be the one this result calls for,

because a program whose interval does not clear its break-even line needs the honest "the evidence does not settle it" sentence of Section 11.7 rather than the confident one.

Then answer three follow-ups. First: the interval's lower bound and the break-even line are close — describe what a decision-maker should be told, and what additional information would settle it. Second: recompute the break-even line at a contribution of $22 rather than $28, state which way the line moves and whether your conclusion changes, and then say which of the two contribution figures you would put in the deliverable and why the other one belongs there too. Third: the CRM lead proposes running the test another two weeks "to tighten the interval." Explain, citing the relevant section, why this is or is not acceptable, and what would make it acceptable.

Exercise 11.4 Design the Test (Required Practice)

For each scenario, produce a complete preregistration using every row of Table 11.3 — including the treatment-dose row, the assignment-mechanism row, and the instrumentation obligation row — and state the unit of randomization with one sentence of justification and one sentence naming the contamination path you are guarding against. For every guardrail you declare, name the field it would be computed from; if that field does not exist, say so and move the guardrail to the instrumentation obligation rather than declaring something you cannot measure.

- StyleCraft wants to know whether adding a free-returns message to the checkout page raises order completion.
- StyleCraft wants to know whether a new in-store styling appointment service raises revenue per visit, and can staff it in at most four stores.
- StyleCraft wants to know whether its paid social spend produces incremental revenue at all, in a market where the platform cannot suppress ads for chosen individuals.
- StyleCraft wants to know whether raising the loyalty program's points-per-dollar rate for Insider members increases their purchase frequency.
- StyleCraft wants to know whether the Drop Alert's effect decays with repeated exposure — the novelty question Section 11.8 says the certified file cannot answer. Specify the design fully: how many drops the assignment is held across, what outcome is recorded after each exposure, what quantity estimates the decay, and what sample size each period-specific comparison would need at 80 percent power to detect a half-point change in the lift.

For the second and third scenarios, state whether a randomized experiment is feasible; where it is not, specify the Section 11.9 design you would use instead, name the comparison group, and state the assumption on which the estimate would rest and the pre-period evidence you would show to support it.

Exercise 11.5 The Full Experiment Deliverable (Homework Submission)

Using the certified ab_test file, produce the complete deliverable described in Section 11.13, as a Colab notebook plus a two-page written report. The notebook must contain, in order and using the chapter's cell numbering as its outline: the data contract of Code 11.3, the assignment-flow table of Code 11.4, the covariate balance profile of Code 11.5 with missing values represented explicitly, the sham-split analysis-path check of Code 11.6, the sample-size and power calculation of Code 11.7 including its sensitivity table, the registered intention-to-treat analysis of Code 11.8, both intervals from Code 11.9 with the rescaling shortcut shown and labeled, the peeking simulation of Code 11.10 reporting all three stopping policies, the single-experiment p-value path of Code 11.11, the break-even comparison of Code 11.12 including its dose-and-contribution sensitivity grid, and the temporal-stability diagnostic of Code 11.13 with its standard errors.

The report must contain the registration with its dose row, the assignment certificate, the primary result in percentage points and in dollars, the relative figure with its own interval, the decision with its rule and its sensitivities, the limitations naming population, window, dose, novelty, and the unmeasured paid social counterfactual, and the guardrail and reversal plan with the instrumentation obligation named. Include a comparison against the known true lift published in the data dictionary's design note, and state in one sentence what that comparison does and does not verify. State explicitly, in one sentence each, which of your exhibits is an A/A test and which is not, and what the multiple of the observational headline over the experimental result actually is —

against the point estimate and against the interval's upper bound, which are different numbers. Groups on Project #1's causal track may submit an extended version of this deliverable; see Appendix F for the rubric.

Exercise 11.6 AI Experiment Audit (Homework Submission)

Run the four-move round trip of Lab 11.2 Part E and document it per Appendix D. Submit: (a) your registration prompt and the assistant's draft, with the UNSPECIFIED rows counted, a note on whether it invented a treatment dose or left it unspecified, and a paragraph on what that count revealed about your own prompt; (b) the deliberately misleading analysis prompt and the assistant's response, audited line by line against Table 8.5's five points and Table 11.7's three, with each failure you found named and quoted — check specifically for a per-protocol drift that silently dropped undelivered customers and for a relative-lift interval built by rescaling a difference interval; (c) the adversarial prompt's output, with the items you kept and the items you discarded, and one sentence of justification for each discard; (d) the assistant's drafted recommendation paragraph, your marked-up version showing every verb you changed and why, and your final version, which must name the tested dose. Your grade rests on the audit, not on the assistant's output: an exchange in which the assistant performed badly and you caught everything is a better submission than one in which it performed well.

11.17.2 In-Class Activities

Exercise 11.7 Spot the Testing Mistake (In-Class Discussion)

Each of the following is a real pattern in commercial testing. Name the failure from Table 11.5, state the mechanism in one sentence, describe what the report would look like, and state the single change that would have prevented it. Two of the items contain a failure Table 11.5 does not list; name those from elsewhere in the chapter.

- "We launched Monday and by Wednesday the treatment was up 11 percent with p = 0.03, so we rolled it out Thursday."
- "Conversion was flat overall, but revenue per customer was up 6 percent and significant, so we're calling it a win."
- "Halfway through we noticed the treatment creative had a typo, so we fixed it and let the test keep running."
- "The new referral offer lifted signups 22 percent in the test. After three months of full rollout, signups are back where they started."
- "The 15-percent-off code we tested with 5,000 customers ended up on a coupon aggregator site in week one, but we kept the test running."
- "We ran the test on our email-engaged segment because they respond fastest, and we're rolling the winner out to the full file."
- "About 8 percent of the treatment arm's messages bounced, so we removed those customers and compared the rest against the full control arm."
- "The interval on the lift was 0.4 to 3.1 points and our break-even is 1.2 points, but the point estimate is 1.75, so we're going ahead."

Exercise 11.8 The Holdout Meeting (In-Class Discussion)

StyleCraft's VP of Marketing proposes a twelve-month holdout: 5 percent of the customer base, chosen at random, will receive no marketing communications at all for a year, so that the total incremental value of the CRM program can finally be measured. The CRM manager objects that the company would be knowingly withholding offers, early access, and loyalty communications from four hundred real customers for a year. The finance partner points out that nobody currently knows whether the CRM program creates value at all, and that the company spends several million dollars a year on it.

Prepare a position. Evaluate the proposal against all four conditions of the "Holdout Is a Debt You Repay" concept box, and state which conditions it meets, which it fails, and how it could be redesigned to meet them all while still answering the question. Address explicitly: how large the holdout actually needs to be given the effect size worth detecting, computed rather than asserted, using the machinery of Code 11.7; whether twelve months is

required or merely convenient; what the customers in the holdout should be told, if anything, and what governs that answer rather than what your instinct says; what repayment would look like at the end; and whether your answer would change if the holdout were selected by a rule — say, lowest-value customers — rather than at random, and what that change would do to both the ethics and the estimate. Then address one further question the chapter raises: this holdout receives *no* communications rather than the existing experience, which is a departure from the bounded-harm condition. State whether that departure is defensible here, and what would have to be true for it to be.

11.17.3 Extensions

Exercise 11.9 Peeking Under Two Stopping Rules (Optional)

Code 11.10 reports three false-positive rates for the same simulated experiments: the final-look rate, the any-significant-difference rate, and the treatment-ahead rate. Extend the simulation in three directions and report what changes. First, vary the number of looks from 2 to 28 and plot all three rates against the number of looks; describe the shape of each curve and explain why the any-direction curve flattens rather than approaching the independent-look bound of $1 - 0.95$ raised to the number of looks. Second, vary the minimum arm size required before a look is taken, from 50 to 1,000, and explain the direction of the effect on the inflation. Third, and this is the one that changes how you read commercial test reports, add a fourth policy: an analyst who stops when the treatment is ahead *and* the observed lift exceeds one percentage point, which is what a business rule with a practical-significance threshold actually looks like. Report its false-positive rate, compare it against the other three, and write one paragraph on whether adding a practical-significance threshold to an undisciplined stopping rule fixes the problem, reduces it, or merely hides it. State a seed for every figure you report, and report the spread across at least five seeds rather than a single decimal.

11.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

A/A test. An experiment in which both arms receive the identical experience, run through the full production assignment, delivery, and analysis pipeline, so that the known-zero true effect grades the machinery rather than any intervention. Because the nominal false-positive rate is not zero, a single significant result is an occasion for investigation rather than a verdict; repeated or systematic failures indicate a defect in randomization, delivery, or measurement (adapted from Kohavi et al., 2013).

Absolute and relative lift. Absolute lift is the treatment-minus-control difference in the metric's own units, in percentage points for a conversion rate, and is the quantity that scales to business volume. Relative lift is that difference as a proportion of the control rate; it is systematically the larger and more persuasive figure whenever the baseline is small, and it transports to a population with a different baseline only under a stated multiplicative-effect assumption. Each requires its own confidence interval, the second constructed as a ratio (adapted from Kohavi et al., 2020).

Break-even lift. The lift at which a program exactly pays for itself: the cost per assigned customer divided by the contribution per incremental conversion. It is undefined until the treatment dose fixes the cost per assigned customer, it belongs in the preregistration's decision rule, and it is compared against the whole confidence interval rather than the point estimate, with its sensitivity to cost and contribution reported beside it (course concept developed for this guide, informed by Kohavi et al., 2020).

Causal inference. The estimation of what an outcome would have been under an alternative exposure, using observed data plus design features or assumptions that make an observed group a credible stand-in for the unobserved counterfactual; its credibility rests on research design rather than on dataset size or model sophistication (adapted from Holland, 1986; Imbens & Rubin, 2015).

Contamination. Influence of one unit's treatment on another unit's outcome — also called interference or spillover — through shared offers, shared inventory, shared households, or market-level effects; it biases the

measured difference and is addressed by enlarging the unit of randomization rather than by post hoc adjustment (adapted from Kohavi et al., 2020).

Counterfactual. The potential outcome that did not occur — the result the unit would have shown under the exposure it did not receive; permanently unobservable, and therefore estimated by comparison across units (adapted from Rubin, 1974; Holland, 1986).

Difference-in-differences. A quasi-experimental estimator that compares the change in outcomes over time in a treated group with the change over the same period in an untreated comparison group, taking the difference of the two changes; valid only under the unverifiable parallel trends assumption, which is supported but not proven by matching pre-period trends (adapted from Card & Krueger, 1994; Angrist & Pischke, 2015).

Evidence hierarchy. An ordering of evidence by the strength of causal claim it supports — anecdote, description and association, model-based association and prediction, quasi-experiment, randomized experiment — whose practical function is to hold a conclusion's language to the strength of the design that produced it. It describes the warrant available from competently executed designs; execution failures can move a study's real warrant below its rung (adapted from Angrist & Pischke, 2015; Kohavi et al., 2020).

Intention-to-treat analysis. Comparison of units according to the condition they were randomly assigned, regardless of whether the treatment was delivered, received, or complied with. It preserves the comparability random assignment created, because groups defined by post-assignment events are no longer randomized; it estimates the effect of being offered the program, which is a different quantity from the effect of the treatment actually received (adapted from Hopewell et al., 2025; Imbens & Rubin, 2015).

Multiple comparisons. The inflation of false-positive risk that occurs when many metrics or segments are tested and the significant ones are reported as though each had been the only test; remedied by declaring one primary outcome in advance and treating unplanned findings as hypotheses for a future experiment (adapted from Simmons et al., 2011; Kohavi et al., 2020).

Novelty effect. A response to a treatment arising from its unfamiliarity rather than its enduring value, producing an initial effect that decays with habituation; its mirror image, primacy, is initial underperformance from disrupted habit. Because both concern change across repeated exposures, detecting either requires outcomes observed over several exposures for the same randomized units rather than a comparison of cohorts assigned at different dates (adapted from Kohavi et al., 2020).

Peeking. Examining an experiment's results before the declared sample size or duration is reached and allowing what is seen to influence stopping or analysis; because each look offers a fresh chance to observe a threshold-crossing fluctuation, it raises the false-positive rate far above the nominal level, undetectably in the final output. The inflation is larger when hits in either direction count and roughly half as large under the realistic policy of stopping only when the treatment is ahead (adapted from Johari et al., 2017).

Potential outcomes. The pair of results a unit would show under treatment and under no treatment, whose difference is the unit's causal effect; exactly one is ever observed, which is the fundamental problem of causal inference (adapted from Rubin, 1974; Holland, 1986).

Preregistration. A dated, unalterable record of an experiment's hypothesis, primary metric, arms, treatment dose, assignment mechanism, unit of randomization, eligible population, sample size, duration, analysis, decision rule, and stopping rule, created before the outcomes are observed; it is what distinguishes a tested prediction from a description of the data that generated it. Every guardrail it declares must be computable from the data the pipeline actually delivers (adapted from Nosek et al., 2018).

Random assignment. Allocation of units to experimental conditions by a chance mechanism independent of every characteristic of the units, making the arms comparable in expectation on all characteristics, measured or not. Under Bernoulli randomization each unit is assigned independently and arm sizes are random; under complete randomization a fixed number of units is drawn and arm sizes are exact. Comparability yields an unbiased effect estimate only when assignment is implemented as planned, outcomes are observed consistently, units are analyzed as assigned, interference is absent, and the treatment is well defined at a stated dose (adapted from Fisher, 1935; Imbens & Rubin, 2015).

Selection bias. The difference between an observed group comparison and the causal effect it is taken to estimate, arising because treated and untreated units differ systematically in ways that affect the outcome; a property of how the groups came to exist, unaffected by dataset size, and directed upward or downward according to the selection mechanism rather than always upward (adapted from Angrist & Pischke, 2009; Imbens & Rubin, 2015).

Sham-split analysis-path check. This guide's name for a diagnostic that divides completed control-arm outcomes at random into two groups and runs the primary test on them. Because the true effect is zero by construction, it grades the metric definition, the test statistic, and the analysis code, and it establishes the nominal false-positive behavior of that path. It grades neither the production randomizer nor delivery nor tracking nor assignment-to-exposure integrity, and it is therefore not an A/A test (course concept developed for this guide, informed by Kohavi et al., 2013).

Statistical power. The probability that an experiment detects an effect of a specified size at a specified significance level, given sample size and outcome variability; required sample size grows roughly with the inverse square of the smallest effect worth detecting, and underpowered tests overstate the effects they do declare significant (adapted from Cohen, 1988; Kohavi et al., 2020).

Treatment and control groups. The arm receiving the intervention under test and the arm receiving the existing experience or declared alternative; the control supplies the estimate of what the treatment arm would have shown absent the treatment, and should represent the decision's real alternative rather than a strawman (adapted from Fisher, 1935; Imbens & Rubin, 2015).

Treatment dose. The number and timing of exposures a treated unit receives, stated as part of the intervention's definition. The dose fixes the cost per assigned customer and therefore the break-even lift; an experiment run at one dose licenses a rollout at that dose, and a rollout at a different frequency is a different intervention whose economics and whose response must both be re-derived (course concept developed for this guide, informed by Kohavi et al., 2020).

Unit of randomization. The entity to which the chance mechanism assigns a condition — customer, session, store, market, or time period — which fixes the effective sample size and the grain the analysis must respect. Outcomes may be recorded more finely, but treatment variation exists only at the assignment grain, and the unit should be at least as large as the unit through which the treatment can reach a person (adapted from Kohavi et al., 2020).

11.19 Further Readings

Students who want additional background may begin with the following readings. The applied treatments are listed first, evidence and methods second, reporting standards last.

- Kohavi et al. (2020) for the standard practitioner's book on controlled experiments — written from a decade of platform-scale practice, and the direct expansion of Sections 11.4 through 11.8. The chapters on metrics, common pitfalls, and organizational maturity are the most useful pages a new marketing analyst can read on this subject, and the treatment of guardrail metrics is where the instrumentation obligation of Table 11.3 comes from.
- Angrist and Pischke (2015) for a genuinely accessible introduction to the logic of causal design, including randomized trials and difference-in-differences — the level above this chapter's Sections 11.2, 11.3, and 11.9 without the econometric prerequisites of their earlier volume.
- Johari et al. (2017) for the definitive practical treatment of peeking — why continuous monitoring corrupts conventional tests, how large the corruption is, and what valid sequential alternatives look like; the paper behind Code 11.10.
- Gordon et al. (2019) for the head-to-head comparison of experimental and observational advertising measurement at scale, and the clearest available evidence for why Section 11.3's warning about statistical adjustment is not a theoretical worry — including the finding that the divergence runs in both directions rather than consistently upward.

- Lewis and Rao (2015) for the power arithmetic of advertising measurement — the argument that many reported advertising returns come from tests that could not have measured them, which is Code 11.7's sensitivity table carried to its industry conclusion.
- Nosek et al. (2018) for preregistration as a general scientific reform, and for the distinction between prediction and postdiction that Section 11.5 borrows wholesale.
- Hopewell et al. (2025) for the CONSORT 2025 reporting guideline, which is where the flow record of Table 11.4 and the intention-to-treat discipline of Section 11.5 come from in their mature form. A marketing analyst who reads the participant-flow and analysis-population items once will never again drop a randomized customer without noticing.
- Senn (1994) and Altman (1985) for the argument that baseline significance tests are the wrong instrument for judging whether a randomization worked — short, pointed, and the direct source of Section 11.5's reframing of the balance check as a diagnostic.

11.20 References

Altman, D. G. (1985). Comparability of randomised groups. *Journal of the Royal Statistical Society: Series D (The Statistician), 34*(1), 125–136. https://doi.org/10.2307/2987510

Angrist, J. D., & Pischke, J.-S. (2009). *Mostly harmless econometrics: An empiricist's companion*. Princeton University Press.

Angrist, J. D., & Pischke, J.-S. (2015). *Mastering 'metrics: The path from cause to effect*. Princeton University Press.

Athey, S., & Imbens, G. W. (2017). The state of applied econometrics: Causality and policy evaluation. *Journal of Economic Perspectives, 31*(2), 3–32. https://doi.org/10.1257/jep.31.2.3

Blake, T., Nosko, C., & Tadelis, S. (2015). Consumer heterogeneity and paid search effectiveness: A large-scale field experiment. *Econometrica, 83*(1), 155–174. https://doi.org/10.3982/ECTA12423

Card, D., & Krueger, A. B. (1994). Minimum wages and employment: A case study of the fast-food industry in New Jersey and Pennsylvania. *American Economic Review, 84*(4), 772–793.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Lawrence Erlbaum Associates.

Fisher, R. A. (1935). *The design of experiments*. Oliver & Boyd.

Gordon, B. R., Zettelmeyer, F., Bhargava, N., & Chapsky, D. (2019). A comparison of approaches to advertising measurement: Evidence from big field experiments at Facebook. *Marketing Science, 38*(2), 193–225. https://doi.org/10.1287/mksc.2018.1135

Holland, P. W. (1986). Statistics and causal inference. *Journal of the American Statistical Association, 81*(396), 945–960. https://doi.org/10.1080/01621459.1986.10478354

Hopewell, S., Chan, A.-W., Collins, G. S., Hróbjartsson, A., Moher, D., Schulz, K. F., Tunn, R., Aggarwal, R., Berkwits, M., Berlin, J. A., Bhandari, N., Butcher, N. J., Campbell, M. K., Chidebe, R. C. W., Elbourne, D., Farmer, A., Fergusson, D. A., Golub, R. M., Goodman, S. N., … Boutron, I. (2025). CONSORT 2025 explanation and elaboration: Updated guideline for reporting randomised trials. *BMJ, 389*, e081124. https://doi.org/10.1136/bmj-2024-081124

Imbens, G. W., & Rubin, D. B. (2015). *Causal inference for statistics, social, and biomedical sciences: An introduction*. Cambridge University Press.

Johari, R., Koomen, P., Pekelis, L., & Walsh, D. (2017). Peeking at A/B tests: Why it matters, and what to do about it. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1517–1525). https://doi.org/10.1145/3097983.3097992

Kohavi, R., Deng, A., Frasca, B., Walker, T., Xu, Y., & Pohlmann, N. (2013). Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1168–1176). https://doi.org/10.1145/2487575.2488217

Kohavi, R., Tang, D., & Xu, Y. (2020). *Trustworthy online controlled experiments: A practical guide to A/B testing*. Cambridge University Press. https://doi.org/10.1017/9781108653985

Kramer, A. D. I., Guillory, J. E., & Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences, 111*(24), 8788–8790. https://doi.org/10.1073/pnas.1320040111

Lewis, R. A., & Rao, J. M. (2015). The unfavorable economics of measuring the returns to advertising. *Quarterly Journal of Economics, 130*(4), 1941–1973. https://doi.org/10.1093/qje/qjv023

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences, 115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology, 66*(5), 688–701. https://doi.org/10.1037/h0037350

Seabold, S., & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference* (pp. 92–96). https://doi.org/10.25080/Majora-92bf1922-011

Senn, S. (1994). Testing for baseline balance in clinical trials. *Statistics in Medicine, 13*(17), 1715–1726. https://doi.org/10.1002/sim.4780131703

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science, 22*(11), 1359–1366. https://doi.org/10.1177/0956797611417632

statsmodels developers. (2026a). *statsmodels.stats.proportion.confint_proportions_2indep* [Software documentation]. https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.confint_proportions_2indep.html

statsmodels developers. (2026b). *Statistics: statsmodels.stats* [Software documentation]. https://www.statsmodels.org/stable/stats.html

Vaver, J., & Koehler, J. (2011). *Measuring ad effectiveness using geo experiments*. Google Inc.

Verma, I. M. (2014). Editorial expression of concern: Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences, 111*(29), 10779. https://doi.org/10.1073/pnas.1412469111

Wasserstein, R. L., & Lazar, N. A. (2016). The ASA statement on p-values: Context, process, and purpose. *The American Statistician, 70*(2), 129–133. https://doi.org/10.1080/00031305.2016.1154108

Chapter 12

Data Visualization for Analysis (Tableau I)

Seeing Before Persuading

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.1 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter builds the analytical half of data visualization, using StyleCraft's store-expansion question as its running material and Tableau as its instrument. It establishes the grammar of a chart — marks, encodings, scales, aggregation, and sort — and the perceptual hierarchy that orders encoding channels by decoding accuracy, converting chart selection from taste into a prediction. Charts are chosen by analytic purpose rather than from a gallery, scales and baselines are assigned by what a mark encodes, and five standing misleading mechanics are demonstrated on StyleCraft's numbers and repaired. Color is treated as an encoding system in which accessibility is a correctness requirement, and reference lines supply Chapter 2's baseline discipline in visual form. A Tableau Prep flow certifies the extract at two grains, six views build the expansion story, and small multiples test whether the newest stores' apparent weakness is largely an artifact of store age.

KEYWORDS

data visualization; perceptual hierarchy; chart selection; visual encoding; axis truncation; color accessibility; reference lines; Tableau; aggregation; small multiples

VERSION AND DATE

Version 1.1 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Data visualization for analysis (Tableau I). In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 12, Version 1.1) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Tableau, Tableau Desktop, and Tableau Prep are trademarks of Salesforce, Inc. Google Colab is a product of Google LLC. ChatGPT is a product of OpenAI. Claude is a product of Anthropic. Gemini is a product of Google LLC. GitHub Copilot is a product of GitHub, Inc. NotebookLM is a product of Google LLC. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

1. Distinguish visualization's analytical job from its communicative job, state which job a given chart was built for, and explain why a chart that succeeds at one can fail at the other.
2. Decompose any chart into its grammar — the marks it draws, the data encoded to each visual channel, and the axes and scales those channels are read against — and describe the chart types used since Chapter 4 in that vocabulary.
3. State the perceptual hierarchy of encoding channels, explain the evidence behind it, and predict which of two candidate encodings a reader will decode more accurately for a stated comparison.
4. Select a chart form from an analytic purpose — distribution, comparison, trend, relationship, composition, or geographic — using a purpose-first selection table rather than a chart-type gallery, and justify the selection in terms of the comparison the reader must make.
5. Choose between linear and logarithmic scales for a stated marketing question, and apply the zero-baseline obligation by chart type rather than as a blanket rule.
6. Recognize, demonstrate, and repair the five standing misleading-chart mechanics — axis truncation, dual axes, cherry-picked windows, area distortion, and three-dimensional effects — and state what each does to a reader's estimate of the underlying quantity.
7. Match a color palette type — sequential, diverging, or categorical — to a variable's role and level of measurement per Section 3.4, and evaluate a palette for contrast and color-vision deficiency as a condition of correctness rather than of politeness.
8. Add reference lines, baselines, and annotation that supply the analytic context a chart cannot otherwise carry, explaining this as the visual form of the baseline discipline introduced in Section 2.7.
9. Map Tableau's dimensions and measures onto the variable roles of Section 3.5, distinguish discrete from continuous fields, and explain how aggregation in the view can silently answer a question at a grain other than the one asked, per Section 3.3.
10. Build a Tableau Prep flow that produces a certified extract at a stated grain, with a row-count and total-revenue reconciliation that verifies the flow did what was intended.
11. Construct core chart forms in Tableau, including small multiples and faceted views, and state what a facet reveals that an aggregate cannot.
12. Audit an AI-drafted chart by predicting its faults from Sections 12.3 through 12.9 before looking at it, correcting it against the Appendix E checklist, and documenting the exchange per Appendix D.
13. Assemble a visual audit of a marketing story that a colleague can reproduce: the question each view answers, the encoding chosen and why, the reference context supplied, the grain the view aggregates at, and the charts considered and rejected.

Chapter 11 closed Part II with an interval and a rule for acting on it. Eleven chapters of machinery now sit behind the analyst as a shelf of finished evidence: a cleaned transaction file with a documented verification log, a

decomposition of the suburban average-order-value gap, four recovered customer segments, a driver model with coefficients in dollars, a scored churn list with a threshold derived from retention economics, a forecast with a prediction interval that survives the store-opening inflection, and an experimentally measured lift with a break-even line beneath it. Every one of those artifacts is currently a table, a metric, or a paragraph — which is to say that every one of them currently exists in a form that only the analyst who built it can read. That is the problem Part III takes up, and it is not a cosmetic one. A finding that cannot be seen is a finding that will be re-litigated by whoever has the most confident slide, and the great majority of analytical work that dies in organizations dies in the last two feet between the notebook and the meeting. Part III therefore treats visualization not as the decoration applied to finished analysis but as an instrument with its own failure modes, its own verification discipline, and its own capacity to produce confident wrong answers. This chapter owns the first of visualization's two jobs — the analytical one, in which a chart is built so that the analyst can find out what the numbers contain, before anyone has decided what the numbers should say. Chapter 13 will own the second. The distinction sounds academic and is entirely practical: the chart that helps an analyst discover the expansion story and the chart that helps a chief executive act on it are different objects, built to different standards, and confusing them is how a room ends up looking at a beautiful picture of nothing.

CONCEPT

What This Chapter Is Really About

A chart is a machine for making a comparison easy. That is the whole of it, and nearly every practical rule in this chapter is a consequence of taking the sentence literally. If the comparison a reader must make is "which of these seventeen stores earned the most," the machine that makes it easy is a sorted bar chart, because human beings compare lengths against a common baseline with high accuracy and compare wedge angles poorly. If the comparison is "did the daily series step up when the store opened," the machine is a line with a dated reference mark, because the eye follows connected position over time and cannot hold nine columns of a table in memory long enough to notice a break. And if a chart is built so that the comparison the reader most needs is the one the encoding makes hardest — categories in a pie, magnitudes in a bubble's area, two unrelated series forced onto twin axes — then the chart has failed at its only job, regardless of how good it looks or how correct its underlying query was. This is why the chapter spends its first half on perception rather than on software. Tableau will draw whatever it is asked to draw, in seconds, beautifully, and an AI assistant will now ask it on the analyst's behalf; neither has any opinion about whether the resulting picture supports the comparison anyone actually needs. The judgment that remains the analyst's is therefore the judgment about what to encode where, which is where every chart is won or lost, and it is a judgment that can be checked in advance. Predict what the reader will see, build it, then look and grade the prediction. That is predict-then-verify, arriving in Part III as the discipline that turns chart-making from a matter of taste into a matter that can be right or wrong.

12.1 Marketing Decision Context: Seeing the Expansion Story

Project #1 was presented last Tuesday. The evidence is assembled, defended, and — for the first time since Chapter 5 — nobody is asking the analyst for another number.

What the VP of Marketing is asking for instead is a pre-read. On December 2, StyleCraft's chief executive convenes a long-scheduled review: whether to commit to eight suburban and resort sites under letter of intent, or to stop the expansion and redirect the capital into the urban and digital business. Flagship candidates remain in the pipeline but are not yet under letter of intent and require a separate economic benchmark; the real-estate committee needs a decision before the leases price, and the decision will be made in one meeting, from one packet, by people

who have not read a line of the analysis behind it. Chapter 13 will build the artifact that meeting runs on. This week's job comes first: the VP wants a visual read of the expansion story — everything Part II established about the suburban and resort cohort, rendered so that a person can see it — circulated to the review's four participants by Monday.

The commission is unusual in one respect worth naming immediately, because it changes what "good" means. Nobody is being persuaded this week. The pre-read's four readers are the analyst's own colleagues — the VP, the finance partner from Chapter 11, the head of merchandising, and the director of real estate — and their job on receipt is to disagree, to notice what is missing, and to arrive on December 2 having already had the argument. A pre-read that makes the expansion look good is a failed pre-read, and so is one that makes it look bad. What is wanted is a set of views in which the story is visible enough to be argued with — a materially different design brief from the one Chapter 13 will answer, and the subject of Section 12.2.

There is also a complication, and it arrived Thursday afternoon in an email with six charts attached. The merchandising analyst — capable, junior, and under the same deadline — asked an AI assistant to "make some charts showing how the suburban stores are doing," pasted the resulting specifications into Tableau, and circulated the result in about ninety minutes. The deck is attractive. It uses the brand's colors, it is titled in complete sentences, and it has already been forwarded to the director of real estate with the note "this is great, let's use these." It also contains, on close reading, four separate errors of the kind this chapter exists to catch, and a fifth that lives in the covering email rather than in any image: the email describes the loyalty-tier chart as showing "average order value," while the worksheet behind it computes `AVG(line_revenue)` on a line-grain extract. The lead chart shows quarterly revenue in the suburban and resort stores rising from $241,000 to $252,000 — a real increase of 4.6 percent — drawn on a vertical axis that begins at $240,000 and ends at $255,000, so that the second bar stands roughly twelve times the height of the first. The second chart puts suburban revenue and cumulative store count on twin vertical axes whose ranges have been chosen, by the software's defaults rather than by anyone's intent, so that the two lines lie almost exactly on top of each other, inviting the reading that each opening produced the revenue that followed it. The third is a pie chart of seven product categories, from which no reader can rank the middle four. The fourth encodes StyleCraft's four loyalty tiers — None, Insider, VIP, and Backstage — in a red-to-green palette that implies an ordering the tiers do have and a valence they do not, and that renders two of the four tiers indistinguishable to the roughly one in twelve male readers with a color-vision deficiency. A distribution list of this size could easily include such a reader, and the design obligation does not depend on knowing whether it does.

None of that was done in bad faith, and the distinction matters for how the analyst handles it. The merchandising analyst did not truncate an axis to mislead the real-estate director; the assistant proposed a chart specification, the software rendered it with a default axis range chosen to fill the available space, and nobody in the chain had a reason to look at the axis. This is the modal way misleading charts enter marketing organizations in 2026. They are not forged. They are defaults, accepted quickly, by people with a deadline and no framework for knowing which defaults are dangerous. Section 12.7 supplies the framework, Section 12.12 supplies the audit that catches it before circulation, and Lab 12.2 makes the student produce and repair the failure herself.

So the commission has two halves, and they are the chapter's two labs. The first is constructive: build the expansion story out of `transactions_enriched`, in six views chosen by analytic purpose, at grains that survive inspection, with the reference context that lets each view be read against something. The second is corrective: take the circulating deck apart against a checklist, repair each chart, and state — in a sentence per chart, for the merchandising analyst rather than for the file — what changed in the reading. And the standard for both is the one the pre-read's purpose implies. Every view must be a view the analyst would be willing to have argued with, which means every view must be built so that a reader who disagrees can see exactly what she is disagreeing with.

12.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The lead chart in the circulating deck is arithmetically correct: suburban and resort revenue really did rise from $241,000 to $252,000. Write one sentence naming what the chart nonetheless communicates falsely, and one sentence explaining why the phrase "but the numbers are right there on the axis" is not a defense.

- The pre-read is meant to be argued with rather than agreed with. List two design choices you would make differently for a chart built to be disagreed with than for a chart built to be accepted, and predict which section of this chapter each choice comes from.
- The dual-axis chart invites the reading that store openings produced the revenue that followed them. Name the chapter and section of Part II that already supplies the correct objection to that reading, and state what the chart would have to show for the objection to be answered.
- Before Section 12.4 tells you, rank these four encodings by how accurately you think a reader can compare two values through them: length of bars from a common baseline, area of circles, angle of pie wedges, position of points along a common axis. Write your ranking down and grade it later.

12.2 Visualization's Two Jobs

The pre-read due Monday and the dashboard due in December are both collections of charts about the same company, built from the same file, in the same software, by the same analyst. They are not the same kind of object, and the reason they are not is the distinction this section installs — a distinction that has been implicit since Chapter 4, when the first histogram was drawn to find out whether the order-value distribution was skewed, and which becomes load-bearing now that Part III has to build both kinds deliberately.

A chart has a dominant job, and it is one of two. In its analytical job, the chart is an instrument of discovery: it is drawn by the analyst, for the analyst, to find out what the data contains, and its success criterion is whether it reveals structure that the analyst did not already know — a break in a series, a bimodal distribution, an outlier store, a relationship that reverses inside a subgroup. In its communicative job, the chart is an instrument of transfer: it is drawn by the analyst, for someone else, to convey a finding already established, and its success criterion is whether a reader who has never seen the data arrives at the intended understanding in the time available. Gelman and Unwin (2013) put the tension precisely, noting that the graphics statisticians build to interrogate data and the graphics designers build to communicate results are optimized against different objectives and routinely disappoint each other's audiences. Tukey (1977) had staked out the first job a generation earlier under the name exploratory data analysis, whose governing image — the analyst as detective, drawing quickly and disposably in order to notice — remains the best description of what an analytical chart is for.

CONCEPT

Analysis Charts and Communication Charts Are Different Objects

An analytical chart is disposable, dense, unpolished, and drawn to answer a question the analyst has not yet answered; twenty of them may be made in an hour and nineteen thrown away. Its labeling is spare rather than absent: a private, disposable view can be terse, but any view that will be revisited, reused, or reproduced still needs units, the filters applied, the grain one mark represents, and field labels a reader can interpret. The difference between the two kinds of chart is density and polish, not whether the chart can be understood later. A communication chart is durable, sparse, finished, and drawn to convey a finding the analyst has already established; there may be four of them in a deliverable, each surviving several rounds of revision, each titled with the sentence it wants the reader to leave holding. The properties that make each good actively harm the other. Density that lets an analyst scan seventeen stores at once overwhelms an executive with ninety seconds; simplification that lets an executive grasp one comparison instantly hides the second comparison the analyst needed to see. Hence the working rule this chapter and the next enforce together: decide which job a chart has before drawing it, and never let a chart drift from the first job to the second without being rebuilt. The two jobs are dominant purposes rather than exclusive categories — some artifacts, the pre-read among them, serve both audiences at once, and naming which job dominates is what makes the design brief decidable. The most common form of that drift is the exploratory view — made in five minutes, full of defaults, correct for its purpose — that gets pasted into a deck because it was already there. This chapter owns analysis charts. Chapter 13 owns communication charts, and its first act will be to rebuild rather than to reuse.

Figure 12.1

Visualization's Two Jobs

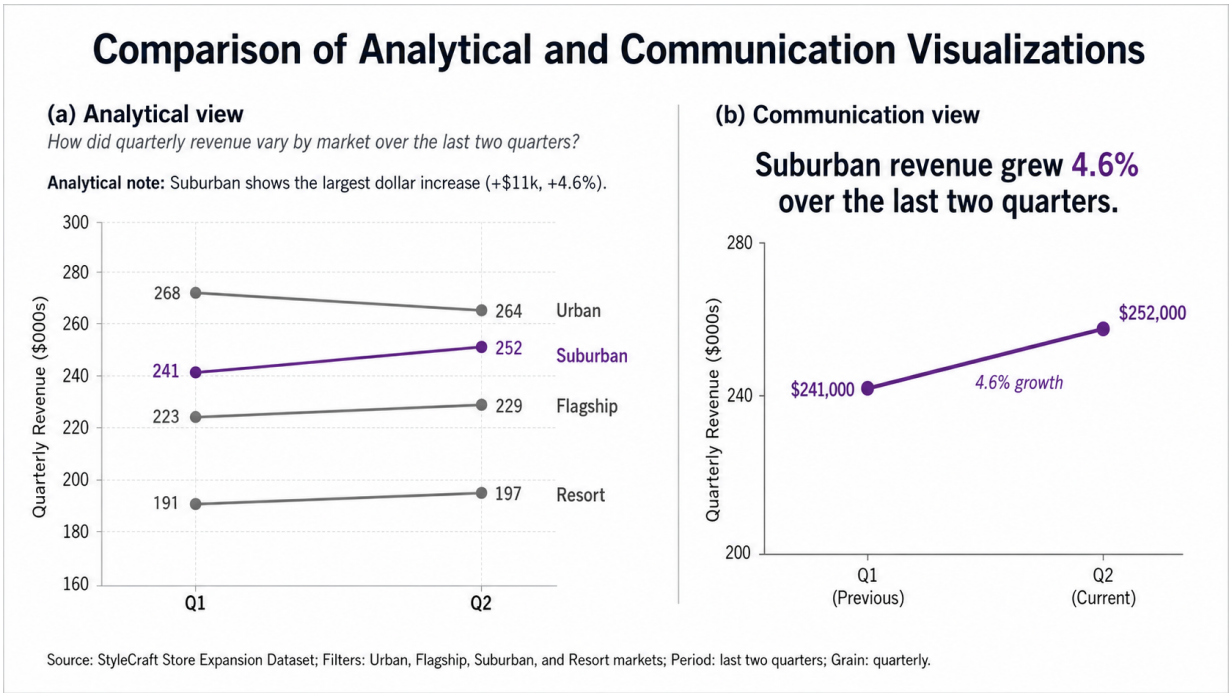


Figure 12.1 sets the two objects beside each other. In other words, the pre-read is an analytical artifact serving a semi-public purpose, which is exactly the awkward middle case that makes it a good teaching problem. Its four readers are colleagues who will interrogate it, so it must carry the density and the honesty of an analytical view; but it leaves the analyst's screen, so it must be legible to someone who did not build it. The resolution the chapter adopts is to hold the analytical standard and add only what legibility requires — real axis labels, units, a note of the grain each view aggregates at, and a reference line where a number needs something to be read against. What it does not add is the assertion titling, the aggressive decluttering, and the narrative sequencing that Chapter 13 will teach, because those techniques direct a reader toward a conclusion, and directing this week's readers toward a conclusion is precisely what the VP does not want.

One consequence of the division deserves stating now, because it governs the rest of the chapter. Analytical charts are where an analyst's own errors are caught, which means the analytical stage is the one that has to be built for self-suspicion rather than for speed. Chapter 4's exploratory charts were read for anomalies; Chapter 7's scatterplots enforced the always-plot discipline that a correlation coefficient alone will not; Chapter 10's decomposition plot made a structural break visible that no summary statistic named. Every one of those was a chart doing the analytical job, and in every case the value came from the chart's willingness to show something inconvenient. A chart built to make a point will not do that, because a chart built to make a point has already decided what it is going to find. The order matters, and it is the order Part III follows: see first, then persuade — and only about what you saw.

12.3 The Grammar of a Chart

Chapter 4 introduced four chart types by name and used them: the histogram for shape, the box plot for spread and outliers, the bar for comparison across categories, and the scatterplot for relationship. They were introduced as objects to be read, and the reading was enough for the job. This section supplies the theory that was deferred — the observation that those four are not four unrelated inventions but four points in a small space of choices, and that

once the space is named, a chart stops being something an analyst selects from a gallery and becomes something she specifies.

The idea is due to Bertin (1983), who cataloged the visual variables available for encoding information on a plane, and to Wilkinson (2005), who formalized the result as a grammar in which a chart is generated by combining a small number of independent components rather than chosen from a fixed list; Wickham (2010) rendered the grammar in the layered form that most modern software implements, Tableau included. The practical payoff for a marketing analyst is not the formalism. It is that a chart can be specified in a sentence, audited component by component, and altered one component at a time.

DEFINITION

Mark and Encoding

A mark is the geometric object a chart draws to represent data — a point, a line, a bar, a rectangle, a circle, an arc, or a filled area — and a chart's mark type determines what kind of thing the reader is looking at. An encoding is the assignment of a data field to a visual channel of those marks: position along an axis, length, angle, area or size, color hue, color intensity, shape, or texture. A chart is fully specified by its marks, its encodings, and the axes and scales against which the positional channels are read; two charts with the same underlying query and different encodings are different instruments answering different questions.

Source: Adapted from Bertin (1983) and Wilkinson (2005).

In other words, when the merchandising analyst's fourth chart maps loyalty tier to color hue, that is an encoding decision, separable from the decision to use bar marks and from the decision to put revenue on the vertical axis, and it can be criticized and replaced without touching either. Take the four chart types of Chapter 4 and read them in this vocabulary. A histogram is a bar mark whose horizontal position encodes a binned continuous variable and whose height encodes a count; the binning is a transformation applied before encoding, which is why bin width changes the picture without changing the data. A box plot is a composite mark whose position encodes distributional summaries — the quartiles, the whiskers, the outlying points — against a single continuous axis. A bar chart of revenue by store is a bar mark whose length encodes revenue and whose categorical position encodes store, with sort order as a design choice that carries real information and is almost always left at the software's default. And a scatterplot is a point mark with two positional encodings, which is why it is the most information-efficient chart in common use and why Chapter 7 insisted on it: both of its channels are the ones readers decode best.

Three components sit underneath the marks and are easy to overlook precisely because software supplies them automatically. The first is the axis and its scale, which determines what a unit of visual distance means and which Section 12.6 takes up in full. The second is the aggregation applied before the mark is drawn: a bar showing "revenue by metro" is showing a sum, and one showing "order value by metro" is almost certainly showing an average, and the two are different claims about the world made with identical geometry — the point Section 12.10 will make operational when Tableau's aggregation in the view is introduced, and the point Section 3.3 established when it observed that changing grain changes the claim. The third is the sort order, which is not decoration. A bar chart of seventeen stores sorted alphabetically and the same chart sorted by revenue contain the same numbers and support different reasoning, because sorting by the encoded value turns the comparison "which is largest" from a search problem into a reading problem.

The reason to hold the grammar in mind rather than a gallery of chart types is that the gallery does not scale and the grammar does. There are perhaps forty named chart types in circulation and an analyst will meet a new one every few months, usually with a vendor's name attached; there are half a dozen mark types and eight or nine encoding channels, and every new chart type is a recombination of them. A radial bar chart is a bar chart with position mapped to angle instead of to a linear axis, which tells you immediately what it costs (Section 12.4 will say why) without requiring any prior opinion about radial bar charts. This is also, and not incidentally, why the grammar is what makes an AI-drafted chart auditable: the assistant returns a specification, the specification decomposes into

marks, encodings, scale, aggregation, and sort, and each component can be checked against the question the chart was supposed to answer.

12.4 The Perceptual Hierarchy

Section 12.3 established that a chart's designer chooses which channel carries which field. This section supplies the fact that makes the choice consequential rather than aesthetic: the channels are not equally good, the ordering among them has been measured, and the ordering is stable enough to be used as a design rule and as a prediction.

The foundational work is Cleveland and McGill (1984), who ran controlled experiments in which subjects judged the ratio between two encoded values and whose accuracy was recorded by channel. Important parts of the resulting ranking have been replicated at scale on crowdsourced platforms with modern stimuli and have held up well (Heer & Bostock, 2010). What that evidence supports is a robust average ordering for elementary quantitative judgments rather than a universal ranking: measured performance varies with the task, the display, the reader, and the context. The ordering is short enough to memorize and stable enough to design against, which is all a working rule needs to be.

DEFINITION

Perceptual Hierarchy

The perceptual hierarchy is the empirically measured ordering of visual encoding channels by the accuracy with which readers extract quantitative values from them. Judgments of position along a common scale are most accurate, followed by position along identical non-aligned scales, then length, then angle and slope, then area, then volume, and finally color intensity and saturation. Because accuracy degrades substantially down the ordering, a quantitative field that must be compared precisely belongs on a high-ranking channel, while low-ranking channels are better used for fields that only need to be grouped, ordered coarsely, or distinguished. Two conditions belong to the definition rather than being caveats appended to it. The ordering ranks decoding accuracy alone, and accuracy is not the only design objective, so a low-ranking channel can be the correct choice for a field that need not be decoded precisely. And the ordering governs the comparison the reader actually makes, which the chart's layout determines as much as its channels do. The hierarchy is therefore a robust average ordering for elementary quantitative judgments, not a universal ranking that overrides a stated analytical purpose.

Source: Adapted from Cleveland and McGill (1984) and Heer and Bostock (2010).

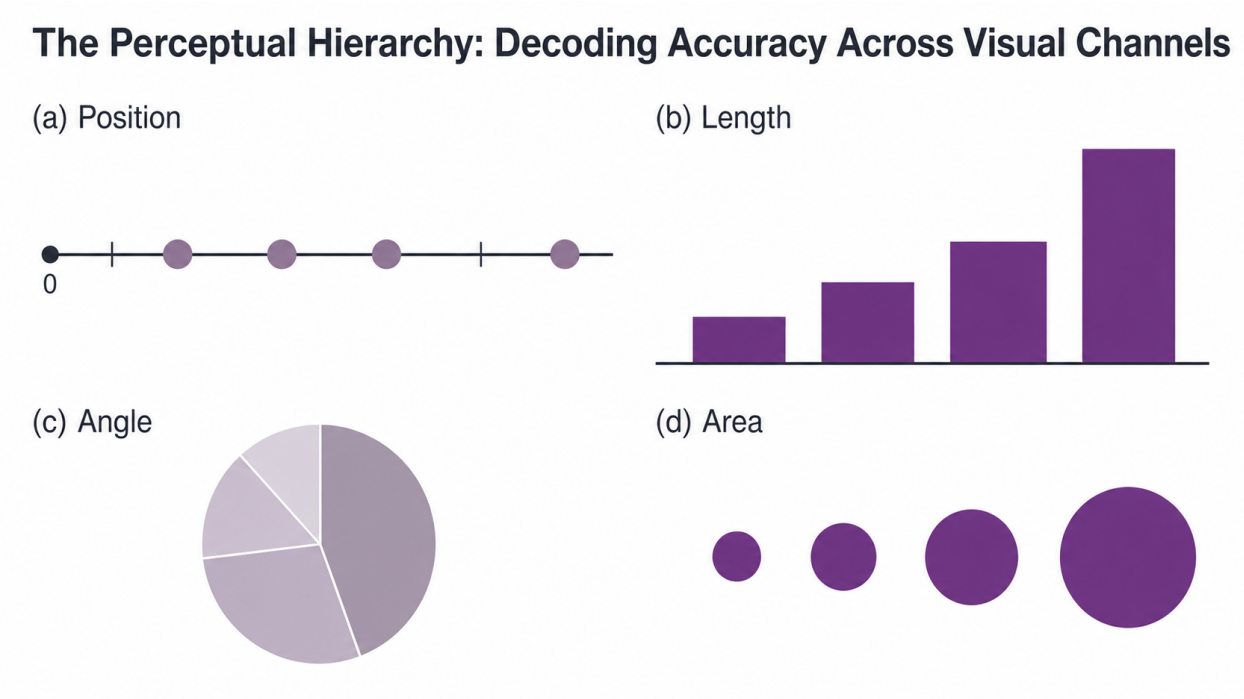
In other words, the eye is a good ruler, a mediocre protractor, and a poor scale for judging area, and a chart's design should spend its best channel on its most important comparison. The consequences are immediate and they explain most of the practical advice in circulation that otherwise sounds like taste. A sorted bar chart beats a pie chart for comparing category magnitudes because bars use length from a common baseline and pies use angle and arc. Bubble maps understate large differences because readers judge circles by something between radius and area and therefore compress the range — which is why a store with four times another store's revenue does not look four times anything. Encoding a quantity by color intensity alone, as a heat map does, is acceptable when the reader needs to spot hot regions and unacceptable when she needs to know whether Rye beat Saddle River. And the reason Chapter 7's scatterplots carried so much weight is now visible: a scatterplot spends both of the top two channels on the two variables whose relationship is the entire question.

Both conditions in the definition deserve a worked instance, because they are what keep the hierarchy from being applied mechanically. On the first, grouping, memorability, and the ability to show many series at once are real objectives, and a channel low in the ordering is the right choice for a field that does not need precise decoding — StyleCraft's four metros carried by color hue on a scatterplot, where the reader needs to tell the groups apart, not

to rank them. On the second, bars in a single sorted column are compared along a common baseline; the same bars split into four side-by-side panels are compared across identical non-aligned scales, one rank lower, which is the small price small multiples pay and Section 12.11 will weigh against what they buy. Figure 12.2 puts the ordering on one set of values so that it can be experienced rather than accepted. Before looking at it, fix the task: in each of the four panels, estimate how many times larger the largest value is than the smallest, and write the four estimates down before reading on. Zacks and Tversky (1999) added a related caution worth carrying: readers infer meaning from mark type itself, reading bars as discrete comparisons and lines as continuous trends, so a line drawn across categorical positions implies a continuity that does not exist, and a bar chart of a time series discards the trend reading the eye is best at.

Figure 12.2

The Perceptual Hierarchy in Practice



The four panels encode the same four values — 8, 14, 23, and 46, so the largest is 5.75 times the smallest. Compare that against the four estimates. Most readers land close on the aligned points and the bars, drift on the wedges, and compress the ratio badly on the circles, which is the ordering the hierarchy predicts and the reason the next section spends its best channel on its most important comparison.

The reason to teach the hierarchy in a chapter about analysis rather than communication is that it converts chart criticism into a prediction, which is what makes it verifiable. Before building a view, the analyst can state what comparison the reader must make, name the channel that comparison will be made through, and predict how accurately it will be made — and then look at the built chart and grade the prediction. That routine is the chapter's verification theme in miniature, it is what Section 12.12 asks of an AI-drafted chart before the chart is even displayed, and it is the first exercise in Section 12.18.

12.5 Choosing Charts by Analytic Purpose

Section 12.4 established which channels carry a comparison well. This section answers the question that actually confronts an analyst at the keyboard, which is not "which channel is most accurate" in the abstract but "what am I trying to find out, and what does that make the right picture." The move is to select from purposes rather than from chart types, and the reason is that a gallery of chart types answers a question nobody has — *what charts exist* — while a purpose list answers the question everybody has, which is *what do I do next*.

DEFINITION

Analytic Purpose

An analytic purpose is the class of question a view is built to answer, stated as the comparison or structure the reader must extract rather than as the chart form that displays it. Six purposes cover nearly all marketing analysis: distribution (how are values of one variable spread), comparison (how do values differ across categories), trend (how does a value move over time), relationship (how do two variables move together), composition (how does a total divide into parts), and geographic (how does a value vary across places). Purpose determines which encoding channels must carry which fields, and therefore determines the chart form; selecting the form first inverts the dependency and produces charts that display data without answering anything.

Source: Adapted from Shneiderman (1996) and Munzner (2014).

In other words, an analyst who opens Tableau thinking "I'll make a bar chart of this" has already skipped the only decision that matters. Amar et al. (2005) cataloged the low-level tasks people actually perform with visualizations — retrieve a value, find an extremum, sort, characterize a distribution, find anomalies, correlate — and the catalog reads as a decomposition of the six purposes into the operations each supports. The practical form of the catalog is Table 12.1, which is written to be used at the moment of building rather than read once. Its rightmost column matters as much as the others: knowing which chart form is wrong for a purpose is what stops an analyst from producing the pie chart in the circulating deck.

Table 12.1

Choosing a Chart by Analytic Purpose

Purpose	The question it answers	A StyleCraft instance from Chapters 4–10	Default form	Common wrong form and why
Distribution	How are the values of one variable spread — center, spread, shape, outliers?	Are suburban order values simply higher, or higher and wider than urban ones (Chapters 4 and 5)?	Histogram; box plot for comparison across a few groups	A bar of the mean, which discards the shape that was the question and hides the bimodality Chapter 5 found
Comparison	How do values differ across categories, and which are largest?	In which store types do returning customers account for the largest share of orders (Chapter 5)?	Sorted horizontal bar	A pie, which replaces length judgments with angle judgments; an unsorted bar, which turns

Purpose	The question it answers	A StyleCraft instance from Chapters 4–10	Default form	Common wrong form and why
				reading into searching
Trend	How does a value move over time, and does the movement break?	Does daily revenue step up at the Greenwich opening (Chapter 10)?	Line, with time on the horizontal axis	A bar chart of a time series, which discards the continuity the eye reads best; a line across categories, which invents continuity that does not exist
Relationship	How do two variables move together, and for which units does the pattern break?	Does app engagement accompany higher spend, and does the pattern hold in both designed segments (Chapter 7)?	Scatterplot, with a fit line where a model exists	A dual-axis time chart, which shows two series against time and is routinely misread as showing them against each other
Composition	How does a total divide into parts, and how does the division change?	How does category revenue split across metros, and is occasionwear's share really the suburban difference (Chapter 5)?	Stacked bar for a few parts; 100 percent stacked bar when shares are the question; treemap for many parts	A pie with more than about five slices, which nobody can rank; a stacked bar used to compare the middle segments, whose baselines float
Geographic	How does a value vary across places, and is location itself the explanation?	Where are the Wave 4 stores, and does the ramp pattern follow geography or store age (Chapters 5 and 9)?	Symbol map sized by value; filled map for rates over defined areas	A filled map of raw counts, which mostly draws a population map; a bubble map used to rank magnitudes precisely

Two rules of use travel with the table. The first is that the purpose is a property of the question, not of the data, and the same fields serve different purposes in different views. StyleCraft's `line_revenue` appears in this chapter as a trend (daily revenue across the openings), a comparison (revenue per store), a composition (category share by metro), and a geographic view (revenue by location), and all four are legitimate because all four answer different questions. What is not legitimate is producing all four and then deciding what the question was, which is the visual form of the multiple-comparisons problem Section 11.8 named and which Section 12.14 will meet again in a different costume.

The second rule is that a purpose can require more than one view, and analysis is where the extra views are cheap. Chapter 5's finding that suburban baskets are larger rather than more expensive is a composition question and a distribution question at once — the mix of occasionwear must be seen and so must the spread of units per order —

and no single chart carries both. The analytical stage's advantage over the communication stage is precisely that it is allowed to answer a question with four pictures. Chapter 13 will have to pick one, and picking one is easier for an analyst who has already seen all four than for one who built only the chart she expected to need.

12.6 Scales and Honest Axes

Marks are drawn against axes, and an axis is a claim about what visual distance means. This section takes up the two decisions that claim involves — whether the scale is linear or logarithmic, and where the axis begins — and it treats them as analytical decisions with correct and incorrect answers for a stated question, rather than as the matters of preference they are usually treated as. Section 12.7 will then show what happens when the same decisions are made badly, which is why the honest treatment comes first: an analyst who cannot say why an axis is where it is has no basis for objecting to anyone else's.

Begin with the scale. On a linear axis, equal vertical distances represent equal differences, so the question the axis answers is "how much more." On a logarithmic axis, equal vertical distances represent equal ratios, so the question it answers is "how many times more" — or, for a series over time, "at what rate is this growing." Neither is more honest than the other; each answers a different question, and the error is using one to answer the other's question. One condition is not a matter of judgment: a logarithmic scale is available only when every plotted value is positive. Zero and negative values cannot be placed on a log axis in Tableau, so a series containing them requires a different transformation or a different chart (Tableau, 2026e).

StyleCraft supplies the case that makes the choice concrete. Plot weekly revenue for the SoHo store, mature and running near $7,900 a week, and for the Washington store, opened four months ago and running near $1,800 a week, on a single linear axis. SoHo's line dominates, Washington's line is a nearly flat trace along the bottom, and the fact that matters most to the December 2 decision — that Washington's weekly revenue has doubled since opening while SoHo's has moved 3 percent — is invisible, not because it is small but because the axis was built to answer a different question. Redraw the same two series on a logarithmic vertical axis and the comparison becomes possible: Washington's line rises visibly steeper than SoHo's, because on a logarithmic vertical scale against linear time the slope of a line represents proportional growth — a straight line indicates approximately constant exponential growth — and two series with very different levels can be compared on the quantity that is actually in dispute. The cost is real and must be stated. Log axes are unfamiliar to most executive readers, gridlines fall at 1, 2, 5, 10 rather than at even intervals, and a reader who does not notice the scale will badly underestimate the level difference. This is why the log axis belongs in an analytical chapter: it is often the right instrument for finding something out and frequently the wrong instrument for telling somebody, and a communication artifact built for a reader with ten minutes and no prior exposure will usually make the opposite call about the same pair of series, for good reasons.

Now the baseline, which is where more damage is done. The rule usually taught — "always start the y-axis at zero" — is wrong as a blanket rule and right for a specific and important class of charts, and the specific class is defined by the perceptual hierarchy rather than by convention.

CONCEPT

The Zero-Baseline Obligation Follows the Encoding, Not the Chart

A bar encodes its value in the *length* of the mark, and length is read as a proportion: a bar twice as long is read as twice as much. That reading is only true if the bar's origin is zero, so a truncated bar chart does not merely exaggerate — it makes a false statement in the reader's most trusted channel. A line or a point encodes its value in *position* against a labeled axis, and position is read as a location rather than as a magnitude, so a line chart's axis may legitimately be restricted to the range where the variation lives, because the question a line answers is usually about movement rather than about level. The obligation, then, attaches to the encoding. An ordinary bar whose full length encodes magnitude must begin at zero. A floating or interval mark — a Gantt bar, a range bar, a waterfall component — measures its length from an explicitly defined reference rather than from zero, and must begin at that reference and say so, because the quantity it encodes is a duration, an interval, or a change rather than a level. For area marks the rule is stated on the area rather than on an axis that may not exist: zero must map to zero area. And any mark whose position carries the value requires a labeled axis whose range was chosen for the question and stated where a reader might be misled. And the analyst's duty in the second case is not a smaller version of the first — a line chart on a restricted axis is honest only if the restriction is visible and defensible, and "the software chose it" is not a defense, as Section 12.7 will show with the deck already circulating.

Table 12.2 turns the principle into the check that runs in Lab 12.2 and in Appendix E.

Table 12.2

Scale and Baseline Obligations by Chart Form

Form	What carries the value	Baseline obligation	Log scale appropriate?	What to state on the chart
Bar or column (magnitude)	Full length from a common origin	Zero baseline required	No — length from zero is meaningless on a log axis	Units and aggregation ("sum of revenue, $000")
Floating, interval, Gantt, or waterfall bar	Length measured from a defined reference	Must begin at the explicitly defined reference, which must be stated	No	The reference the length is measured from
Stacked bar or area	Length of each segment; total height	Zero baseline required	No	Whether segments are values or shares
Line	Position against the axis	Not required; range chosen for the question	Yes, when growth rates are the question	The axis range, and the scale type if not linear
Scatterplot	Position on two axes	Not required on either axis	Yes on either axis, when the relationship is multiplicative	Units on both axes; what one point is
Dot plot	Position against a category axis	Not required; label the value axis clearly	Yes	The value axis range

Form	What carries the value	Baseline obligation	Log scale appropriate?	What to state on the chart
Bubble or symbol map	Area of the mark	Area proportional to value, with zero mapped to zero area	Rarely; only with a stated legend	A size legend with at least two labeled reference sizes
Index or percent-change view	Position relative to a base period	The base (100 or 0 percent) must be drawn as a reference line	Sometimes, for compounding growth	The base period, explicitly

One further point closes the section, because it is the one most often missed in Tableau specifically. Software chooses axis ranges to fill the available space, which means the default behavior of nearly every charting tool is to truncate. Tableau's default for a continuous measure on a bar chart is to include zero, which is correct; its default for a line chart is a range fitted to the data, which is defensible; and the moment a filter is applied, a measure is switched, or a quick table calculation is added, the range recomputes silently. An axis is therefore not a thing an analyst sets once. It is a thing an analyst checks every time the view changes, and the check takes two seconds — read the bottom number on the axis and ask whether a reader would guess it.

12.7 How Charts Mislead

Everything in Sections 12.3 through 12.6 was stated positively. This section states the same material as a catalog of failures, because the failures are what an analyst actually encounters — in a vendor's case study, in another team's deck, in an AI assistant's output, and, most instructively, in her own first drafts. Five mechanics account for most of the damage. Each is a specific misuse of a specific component of the grammar, each leaves a recognizable signature, and each has a repair. The mechanics are owned here; the professional obligation they create — that an analyst who can recognize them is the last line before a distorted chart ships — belongs to Section 12.16 and is not argued in this section.

The first mechanic is the one in the circulating deck's lead chart.

DEFINITION

Axis Truncation

Axis truncation is the omission of the zero origin from an axis whose marks encode value by length or area, so that the visual proportion between marks no longer corresponds to the numerical proportion between the values. Because readers decode bar length as a ratio, truncation inflates apparent differences by a factor that depends on how far the axis floor sits above zero and that can be arbitrarily large; the distortion is invisible to a reader who does not inspect the axis, and it survives the presence of correct numerical labels. Truncation is a legitimate choice for position-encoded marks such as lines and points, where it restricts the view rather than misstating the proportion.

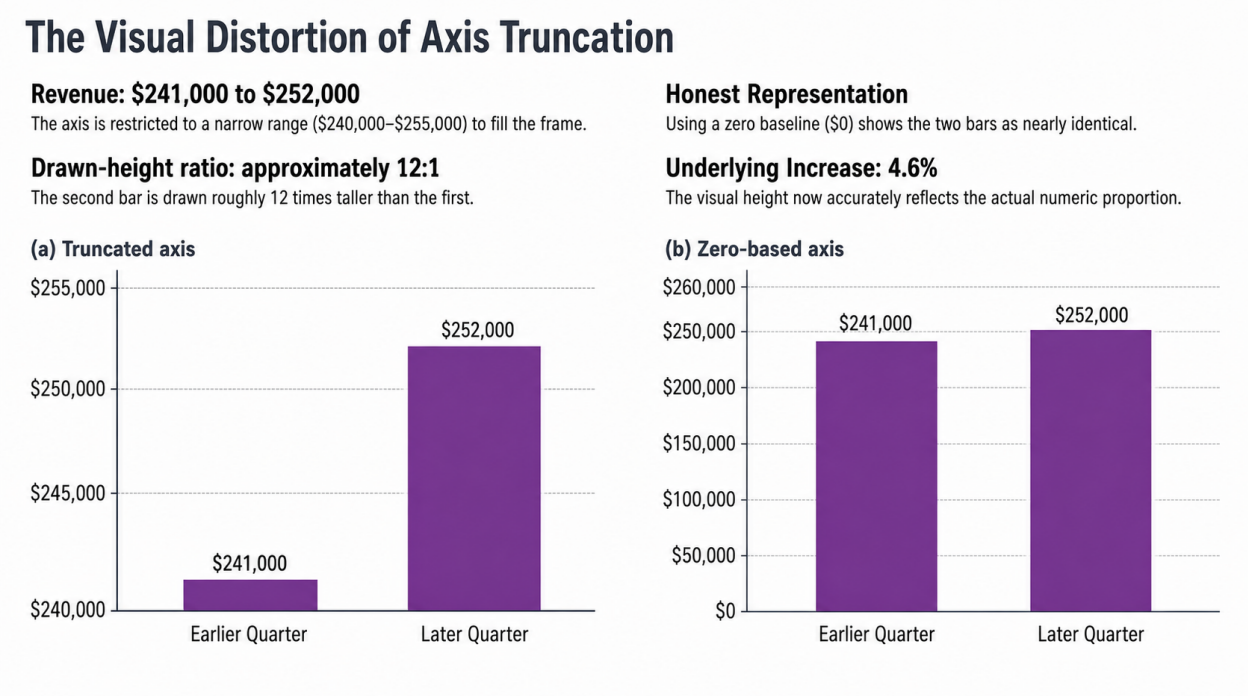
Source: Adapted from Pandey et al. (2015) and Correll et al. (2020).

Work the deck's chart arithmetically, because the size of the effect is the argument. Quarterly revenue in the suburban and resort stores moved from $241,000 to $252,000, an increase of 4.6 percent. Drawn on an axis running from $240,000 to $255,000, the first bar occupies about 7 percent of the plot's height and the second about 80 percent. State the effect in the two quantities that are actually comparable, and avoid any single "exaggeration factor" whose numerator and denominator are undeclared: the second bar is drawn roughly twelve times as tall as the first even though the underlying value is only 1.046 times as large, and a reader who converts the drawing into percentage growth sees an apparent increase of roughly 1,100 percent against an actual increase of about 4.6 percent. The misstatement is made in the channel the perceptual hierarchy says readers trust most, while the axis

labels displayed are entirely correct. Pandey et al. (2015) measured exactly this effect experimentally and found that truncated axes substantially shifted readers' judgments of magnitude relative to identical data drawn from zero. Correll et al. (2020) added the finding that matters for practice: the distortion persists even when readers are shown the axis clearly and even when they are warned, which is why "the numbers are right there" is not a defense and why the repair has to be structural. The repair is to draw the bars from zero — at which point, as Figure 12.3 shows, the two bars look nearly identical, which is the honest picture of a 4.6 percent change — and, if the small change is genuinely the finding, to switch the encoding rather than the axis: a line chart of the eight quarters on a restricted, labeled axis, or a bar chart of quarter-over-quarter percentage change drawn from a zero reference line, both of which put the small movement on a positional channel where restricting the range is legitimate.

Figure 12.3

Axis Truncation Changes the Reading



The second mechanic is the deck's second chart, and it is subtler because it usually has no dishonest intention behind it at all.

Definition

Dual Axis

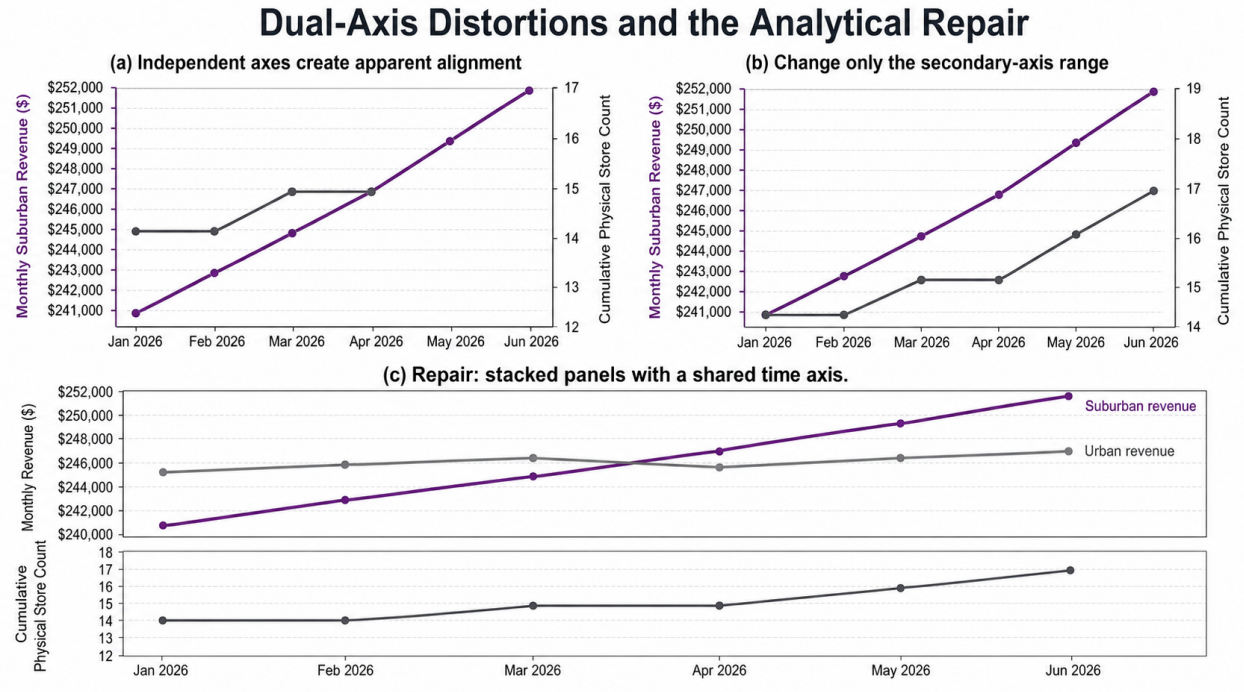
A dual-axis chart plots two measures against a shared horizontal axis using two independent vertical scales, one on each side. Because each scale's range can be set independently and the relative position of the two series depends entirely on that arbitrary choice, the apparent alignment, divergence, or crossing of the series is a property of the scaling rather than of the data; the same two series can be made to appear tightly coupled or unrelated by adjusting either range. Dual axes are defensible only when the two measures are genuinely related quantities whose scales are fixed by definition, and are otherwise best replaced by two vertically stacked panels sharing the horizontal axis, or by indexing both series to a common base.

Source: Adapted from Cairo (2019) and Few (2012).

The deck's chart puts monthly suburban revenue on the left axis and cumulative store count on the right, with both ranges chosen by the software to fill the frame, and the two lines run nearly on top of each other. The reader's inference — each opening produced the revenue that followed — is invited by the geometry and supported by nothing. Two objections are available and the analyst should have both ready. The first is Section 12.6's: the coincidence is manufactured by two independent range choices, and shifting the right axis by two stores separates the lines entirely without changing a single data point. The second is Part II's, and it is the more important one: even a real coincidence between the two series would be a correlation in observational data with a season and a national brand trend running underneath it, which is the confounding of Section 7.5 and the reason Chapter 11 exists. Figure 12.4 makes the dependence on the axis setting visible and shows the repaired form. The repair is to stack two panels sharing a time axis so that each series is read on its own honest scale, to add the urban series as the comparison group whose movement over the same period is what the suburban series must be judged against, and to leave the causal question to the design Section 11.9 sketched rather than to the drawing.

Figure 12.4

Dual Axes Manufacture Relationships



The third mechanic requires no misuse of any axis at all.

Cherry-picking is the selection of the displayed window — of time, of categories, or of units — so that the visible pattern differs from the pattern in the full data. The deck does not contain this one, but the version the analyst will meet before December certainly does: an eight-week window beginning at the Greenwich opening, over which suburban revenue rose 31 percent. Every number in that chart is correct. What the window omits is that the eight weeks run from late October into the holiday peak that Section 10.3's decomposition attributes to seasonality, and that suburban revenue had also been rising through the six months before the opening. Extend the window to the full twenty-four months and the step at the opening is still there — Chapter 10 found it and priced it — but it is a modest step on a rising, strongly seasonal series rather than the takeoff the short window shows. The signature of the mechanic is a window whose start date has no stated justification, and the repair is a standing rule rather than a case-by-case judgment: state why the window starts where it starts, and show at least one full seasonal cycle whenever the series has a season, per Section 10.3.

The fourth mechanic is a failure of the encoding rather than of the scale.

DEFINITION

Area Distortion

Area distortion is the misencoding of a quantity in the size of a two-dimensional mark, most commonly by scaling the mark's radius or side length in proportion to the value so that its area grows with the square of the value. A circle drawn with twice the radius covers four times the area and is read as far more than twice as much; the resulting exaggeration compounds with the general tendency of readers to underestimate area differences, so that size-encoded quantities are simultaneously overstated by the construction and imprecisely decoded by the eye. Correct practice scales area in proportion to value and supplies a size legend with labeled reference marks; precise magnitude comparisons should be moved to a length- or position-encoded view.

Source: Adapted from Cleveland and McGill (1984) and Tufte (2001).

StyleCraft's store map is where this arrives. Trailing-twelve-month revenue runs from roughly $412,000 at SoHo to roughly $26,000 at Washington, which has been open four months and has ramped from under $1,000 a week to $1,800, a ratio of about 16 to 1. Encoded correctly by area, SoHo's circle has about four times Washington's diameter, which already produces a mark large enough to cover several neighboring stores; encoded by radius, it would have 16 times the diameter and roughly 250 times the area, which is not a chart but a blot. The deeper point is that neither version supports the comparison anyway, because Section 12.4 ranks area low. The repair is a division of labor between views: use the map for what maps are uniquely good at, which is showing where things are and whether spatial adjacency explains anything, and put the magnitude comparison in a sorted bar chart beside it. An analyst who wants both from one picture will get neither.

The fifth mechanic can be dealt with briefly, because it has no legitimate use in analysis. Three-dimensional effects applied to two-dimensional data — extruded bars, tilted pies, perspective ribbons — add a depth channel that carries no information and actively corrupts the channels that do: perspective makes near marks larger than far marks of equal value, occlusion hides small marks behind large ones, and the reader has no reliable way to determine where a bar's top intersects the axis. Wainer (1984), writing a deliberately inverted guide to displaying data badly, cataloged the family; nothing since has rehabilitated it. The repair is to remove the effect, and the practice lesson is that a chart type offered by the software is not thereby endorsed by anyone.

Table 12.3 collects the five for use rather than for reading. As with Table 11.4, the symptom column is the one that gets used most, because in practice the analyst is looking at somebody else's finished chart with no access to the underlying view.

Table 12.3

The Five Standing Ways Charts Mislead

Mechanic	What is misused	Symptom in a finished chart	Repair
Axis truncation	The baseline of a length-encoded mark	Bars whose height ratio is wildly larger than their labels' ratio; an axis whose lowest gridline is a suspiciously round non-zero number	Draw length-encoded marks from zero; move a genuinely small difference to a line or a percent-change view with a labeled reference
Dual axes	Two independent scale ranges on one frame	Two series that track each other implausibly well; a right-hand axis	Stack two panels on a shared time axis, or index both series to a

Mechanic	What is misused	Symptom in a finished chart	Repair
		whose range has no natural relationship to the left	common base; add the comparison series the causal reading requires
Cherry-picked window	The extent of the displayed data	A window with no stated start rationale; a period shorter than one seasonal cycle in a seasonal business	State the window's justification; show at least one full cycle; display the pre-period whenever an event is being credited
Area distortion	The mapping from value to mark size	Size differences that dwarf the value differences; a size legend with one entry or none	Scale area, not radius, in proportion to value; supply a two-point size legend; move precise comparisons to a bar chart
Three-dimensional effects	Depth as a decorative channel	Extruded bars, tilted pies, perspective; marks occluding one another	Remove the effect; there is no analytical case for it

One closing observation belongs here rather than in Section 12.16, because it is a fact about how these charts get made rather than a claim about anyone's character. Four of the five mechanics are available as software defaults or one-click options, and the fifth — the cherry-picked window — is what a date filter does when nobody chooses a date. That is the reason this catalog is taught to analysts rather than to executives. The people in a position to catch these are the people whose hands are on the tool, and catching them requires knowing what to look at, which is a checklist, which is Appendix E.

12.8 Color as an Encoding System

Section 12.7 cataloged the failures of the positional channels and the marks drawn against them; this section takes up the channel that catalog barely touched, and it is the one where the most damage is done quietly. Color is the channel most used and least designed. It arrives in every chart whether or not anyone chose it, it is the first thing a reader notices and the last thing a builder inspects, and it is the only encoding channel in common use that can render a chart unreadable for a substantial share of the audience without anyone in the room being aware of it. This section treats color as what Section 12.3's grammar says it is — a set of visual channels to which fields are assigned — and derives the palette rules from the kind of field being assigned, which means from the levels of measurement introduced in Section 3.4.

DEFINITION

Sequential, Diverging, and Categorical Palettes

A sequential palette varies a single hue through a monotonic progression of lightness or saturation and encodes a quantity with a meaningful low-to-high ordering, such as revenue or conversion rate. A diverging palette runs two contrasting hues outward from a neutral midpoint and encodes a quantity with a meaningful center — a target, a zero, a chain average — where distance and direction from that center are both the point. A categorical (or qualitative) palette assigns distinguishable hues of roughly equal lightness to unordered groups, and carries no magnitude information at all; beyond roughly seven or eight categories color alone usually becomes an unreliable identifier and should be supplemented with direct labels or replaced by a different channel — a practical working limit rather than a fixed perceptual boundary. Matching palette type to the field's level of measurement is a correctness requirement: a categorical field in a sequential palette asserts an order the data does not have, and a quantitative field in a categorical palette discards the order it does.

Source: Adapted from Harrower and Brewer (2003) and Munzner (2014).

In other words, the palette is chosen by asking what kind of variable is being colored, which is a question Chapter 3 already taught students to answer. Table 12.4 completes the mapping.

Table 12.4

Palette Type by Variable Role and Level of Measurement

Field being colored	**Level of measurement (Section 3.4)**	**Palette**	**StyleCraft example**	**Failure if mismatched**
Unordered groups	Nominal	Categorical, equal lightness, no more than 7–8 hues	`home_metro`; `acquisition_channel`; `channel`	A sequential ramp implies NYC Urban is "less" than Resort
Ordered groups	Ordinal	Sequential, light-to-dark within one hue	`loyalty_tier` (None → Insider → VIP → Backstage); `age_band`	A categorical palette discards the tier order the business cares about
Quantity with no meaningful center	Ratio or interval	Sequential	Revenue per store; units per order	A diverging palette invents a midpoint and splits the range into two false classes
Quantity measured against a target or a zero	Ratio or interval, with a reference	Diverging, midpoint set at the reference	Contribution margin versus the chain average; forecast error versus zero	A sequential ramp buries the sign change that is the whole finding
A binary flag used for emphasis	Nominal, two values	Two colors: one accent, one neutral gray	`is_occasionwear`; treatment versus control from Chapter 11	Two saturated hues make both groups shout and neither stand out

Figure 12.5

Color Is an Encoding System

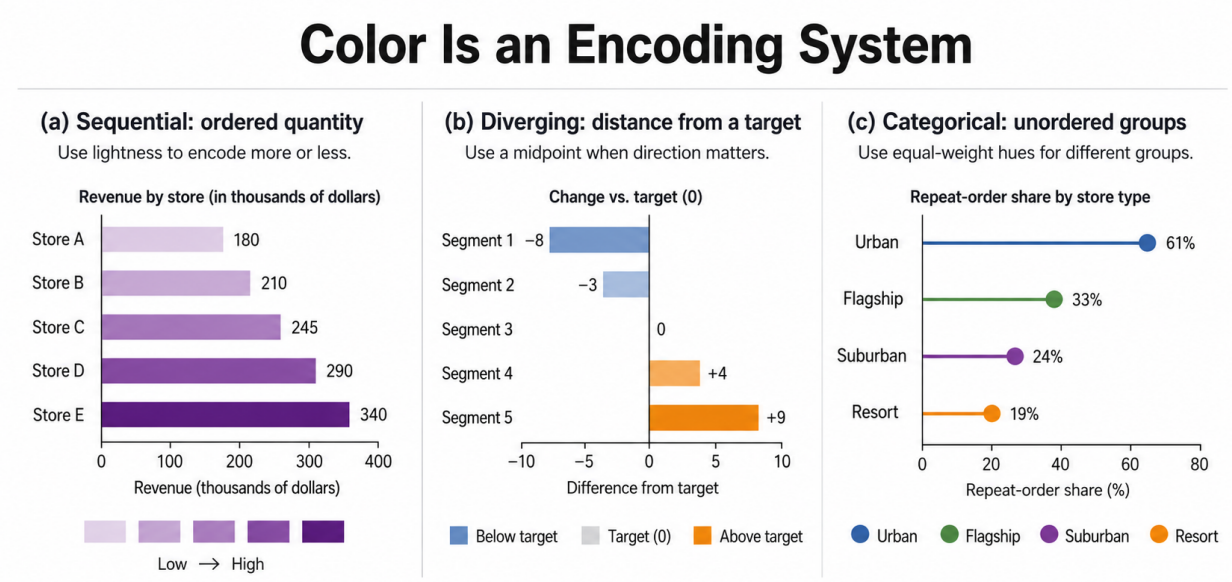


Figure 12.5 shows the three palette types applied correctly and summarizes the accessibility checks each visualization must undergo. Three rules of construction do most of the practical work, and all three are violated by the fourth chart in the circulating deck.

The first is that color must not be the only channel carrying a distinction the reader needs. Roughly 8 percent of men and 0.5 percent of women of Northern European descent have some form of red–green color-vision deficiency, and the deficiency is invisible to everyone else in the meeting, including the person who chose the palette (Wong, 2011). The deck's red-to-green loyalty ramp fails twice over: it uses the two hues that some readers confuse most, and it uses hue alone to separate categories that a light-to-dark progression within a single hue would separate for everyone. The general repair is redundant encoding — carry the distinction in position, in ordering, in direct labels, in shape or line pattern, or in lightness as well as hue — and direct labels, position, and ordering are better safeguards than any choice of hue. The test runs in three parts, and the first part alone is a screening check rather than a certification. Convert the chart to grayscale: if the categories collapse, the chart depends on hue alone and must be rebuilt, and if they survive, the chart has passed one test and not all of them, because grayscale exposes reliance on hue and weak lightness contrast but does not simulate every form of color-vision deficiency. Then inspect the chart through a color-vision-deficiency simulator. Then confirm that color is not the only means by which any essential meaning is carried, which is the requirement accessibility guidance actually imposes, along with adequate non-text contrast for graphical elements. Contrast obeys the same logic. Light gray text on white, and pale categorical hues on a light background, fail for readers with ordinary vision on a projector in a lit room, which describes the December 2 meeting exactly.

The second rule is that color should be spent, not spread. A chart in which every category is a different saturated hue has used its most attention-grabbing channel to encode the least interesting fact, which is that the categories are different from one another — something the axis already said. The high-value use of color in an analytical chart is emphasis: one accent hue for the series under examination, gray for everything else, so that the eye goes where the question is. StyleCraft's seventeen-store weekly revenue chart is unreadable in seventeen hues and immediately readable with the three suburban stores in one accent color and the other fourteen in gray, and no information is lost, because the store identities live in the labels where they belong.

The third rule is that a palette applied to a quantity must be interpretable as a scale, which means it needs a legend with real values and a progression the eye reads monotonically. The classic violation is the rainbow ramp, still the default in some tools and still popular because it is colorful: it is not perceptually uniform, so equal steps in

the data produce visually unequal steps; it introduces sharp apparent boundaries at yellow and cyan where the data has none; and it is unreadable in grayscale and under most color-vision deficiencies (Borland & Taylor, 2007). A single-hue sequential ramp answers every question the rainbow was supposed to answer and answers them correctly.

Finally, a note on brand palettes, because it is the question students ask first and the one that causes the most trouble in practice. StyleCraft has brand colors, and marketing charts are frequently expected to use them. Brand palettes are designed for identity rather than for encoding, so they typically contain three or four hues of similar lightness and no sequential ramp at all, which makes them adequate for a two-or-three-category chart and unusable for anything quantitative. The professional resolution is not to fight the brand and not to bend the data: use the brand hues for accent and identity, derive a sequential ramp by varying lightness within a single brand hue, and keep a neutral gray as the workhorse. Where a chart's encoding requirement and the brand guideline genuinely conflict, the encoding wins in an analytical chart and the conversation gets escalated for a communication one — a boundary an analyst has to be able to defend in front of an actual audience. What no palette can supply, however sensibly chosen, is something to read the values against, which is the subject of Section 12.9.

12.9 Reference Points, Baselines, and Annotation

A chart with a correct encoding, an honest axis, and a defensible palette can still be uninterpretable, and the reason has nothing to do with any of them. A number displayed alone answers "how much" and leaves the reader with the question she actually has, which is "compared to what." This section supplies the answer on the canvas, and it is the direct visual descendant of a discipline this guide installed in Chapter 2: the baseline as the thing a result must be read against, introduced in Section 2.7 as an interpretive obligation, operationalized in Section 8.7 as the model a new model must beat, extended in Section 10.5 to the naive and seasonal-naive forecasts, and rendered here in ink.

DEFINITION

Reference Line

A reference line is a line, band, or marker drawn at a separately defined benchmark — external or data-derived — against which the plotted marks are interpreted. The benchmark may be a target, an average, a median, a prior period, a break-even level, a threshold, or a dated event. Its function is to supply the comparison the reader would otherwise have to supply from memory, converting an absolute display into a relative judgment; a reference band performs the same role for a range rather than a point. References come in two kinds, and the discipline differs between them. An externally supplied decision threshold — a target, a break-even level, a committed plan number — must be declared before the results are inspected, because a threshold chosen after the fact is not a benchmark but a description. A descriptive reference such as an average, a median, or a prior period may legitimately be computed from the data being plotted, provided it is labeled as such so that a reader knows the line is a summary of the marks rather than an independent standard. In both cases the reference must be named on the chart, because the choice of reference determines what the chart appears to say.

Source: Adapted from Few (2012) and Munzner (2014).

StyleCraft's expansion story needs four of them, and naming them is a useful demonstration of how much interpretive work the ink does. A bar chart of average order value by home metro needs a reference line at the chain average, because "$156 for NYC Suburban customers" means nothing until the reader knows the chain runs near $108. A line chart of daily revenue needs a dated vertical reference at the Greenwich opening, because the analytical question is whether the series stepped at that date and the eye cannot locate a date on a two-year axis unaided — this is the structural break of Section 10.4, drawn. A bar chart of contribution by store type needs a reference at the level that covers a store's fixed footprint, because the question the real-estate committee is actually asking is whether

these stores clear their own cost. And a chart of repeat-purchase rate by store cohort needs a reference at the rate the urban cohort reached at the same age, which is the comparison-group logic of Section 11.9 doing its work in a picture rather than in an estimator.

CONCEPT

A Chart Without a Reference Is a Number Without a Baseline

Section 2.7 established that a result quoted without a baseline is not evidence but a quantity, and every chapter since has honored the rule in prose: the model beat a stated baseline, the forecast beat seasonal-naive, the treatment arm beat a control that represented the real alternative. The visual form of the same rule is that a chart which displays values and supplies nothing to read them against has offloaded the interpretive work onto a reader who will do it from memory, from expectation, or from whatever the last chart showed — and will do it badly and invisibly. The working rule this chapter adopts, and Appendix E checks: before a view leaves the analyst's screen, name the comparison the reader must make and confirm that the thing being compared against is *on the chart*. If the comparison is against a target, draw the target. If it is against last year, plot last year. If it is against the other stores, show the other stores or the average of them. And if the honest answer is that there is nothing to compare against — which happens, and is worth saying — then the chart is a description and its title should not contain a word like *strong*, *weak*, *up*, or *behind*, because all four are comparative claims that the chart has not earned.

Annotation is the second instrument, and it does a different job.

DEFINITION

Annotation

An annotation is explanatory text placed on the chart itself — a label on a specific mark, a note on a region, a caption on an axis, or a statement of the view's grain, filters, and source — that supplies context the encoding cannot carry. Annotation is how a chart records what its builder knows and its reader does not: why a point is anomalous, what happened on a marked date, which units are excluded, and at what grain the values are aggregated. Because it travels with the image, annotation is also the mechanism by which an analytical chart remains interpretable after it is separated from the analyst who built it.

Source: Adapted from Tufte (2001) and Few (2012).

Two habits make annotation useful rather than decorative. The first is to annotate the exception rather than the rule: the reader can see that the series rises, and cannot see that the November spike sits inside a promotional window recorded in the `campaigns` table, or that the small step in a March is the near-duplicate re-export that Chapter 4's defect catalog documents and that this extract removed. Those are exactly the facts that belong in ink on the chart, and they are exactly the facts that live only in the analyst's head. The second is the footer discipline, which is unglamorous and is the single highest-return habit in this chapter: every analytical view carries, in small type, the source table, the date range, the filters applied, the grain of aggregation, and the date the extract was refreshed. Five lines of text, added once to a template, prevent the most common failure of the pre-read genre — a chart that circulates for three weeks while nobody can determine whether it includes online orders.

One boundary should be marked before the section closes, since both this chapter and the next put words on charts: annotation supplies the context a reader needs in order to interpret correctly, while the titling discipline Chapter 13 owns supplies the conclusion the analyst wants her to take away, and the pre-read due Monday carries

the first and deliberately withholds the second, for the reason Section 12.2 gave. With the canvas now carrying its own context, the chapter can turn to the tool that will draw it.

12.10 The Tableau Data Model

The chapter has so far described charts in a vocabulary that belongs to no particular software. This section takes up Tableau specifically, and the argument it makes is that Tableau's central abstraction is closely analogous to a distinction students already own from Chapter 3 — which is why an analyst who understood Section 3.5 will find Tableau's behavior largely predictable, and one who did not will find it arbitrary. The analogy is close but not an identity: dimensions and measures are software roles governing how Tableau builds a query, not the analytic variable roles of Section 3.5, and the two can come apart. A field can be converted from measure to dimension or back, and role is orthogonal to whether a field is discrete or continuous, so any field may occupy one of four states — discrete dimension, continuous dimension, discrete measure, or continuous measure (Tableau, 2026a).

DEFINITION

Dimension and Measure

In Tableau's data model, a dimension is a field used to partition, group, or slice the data — it defines the categories, dates, and identifiers at which values are separated — while a measure is a field that is aggregated within those partitions to produce a number. The distinction is closely analogous to the variable roles of Section 3.5: dimensions are typically the categorical and identifying fields that define the units and groups of an analysis, and measures are typically the quantitative fields from which metrics are computed. It remains a software role rather than an exact equivalent — dimensions affect the level of detail of the view, measures are aggregated by default, and a field's role can be changed without changing what the field means (Tableau, 2026a). Because Tableau aggregates every measure to the level defined by the dimensions present in the view, the dimensions in a view determine what each mark represents, and adding or removing one changes the question the chart answers.

Source: Adapted from Murray (2016) and the variable-role framework of Section 3.5.

Two further distinctions govern how fields behave once placed, and they are the ones that confuse new users because they are orthogonal to the first.

DEFINITION

Discrete and Continuous Fields

A discrete field takes a finite set of separate values and produces headers — labeled panels, categories, or bins along an axis — while a continuous field takes values along an unbroken range and produces an axis. In Tableau the distinction is displayed by color, with discrete fields shown in blue and continuous fields in green, and it is independent of whether a field is a dimension or a measure: a date can be treated discretely as a set of month labels or continuously as a timeline, and a measure can be made discrete to produce labeled panels rather than an axis. The choice determines whether the chart's marks are separated into panels or positioned along a scale, which is a decision about the chart's grammar rather than about the data.

Source: Adapted from Murray (2016).

DEFINITION

Aggregation in the View

Aggregation in the view is Tableau's behavior of computing each measure at the level of detail defined by the dimensions currently placed in the view, so that the same underlying field yields different numbers as dimensions are added or removed. Because the aggregation is performed silently and recomputed on every change, a view's values are a property of its dimension set rather than of the measure alone; a metric whose definition requires a specific grain — an average per order, per customer, or per store — will be computed at the wrong grain, without warning or error, whenever the view's dimensions do not match that grain.

Source: Adapted from Murray (2016) and the grain discipline of Section 3.3.

The last of those three is where the chapter's most expensive error lives, and StyleCraft's own headline finding demonstrates it with unusual clarity. `transactions_enriched` is a flat file at order-line grain: one row per line of one order. Drag `line_revenue` into a view, set the aggregation to Average, split by `home_metro`, and Tableau returns the average line value — roughly $49.60 for NYC Urban customers and $51.10 for NYC Suburban, a difference of about 3 percent. That number is computed correctly, and it is not average order value. Average order value requires the order grain, which in this file means the sum of revenue divided by the count of distinct orders: `SUM([line_revenue]) / COUNTD([order_id])`, which returns roughly $94 for urban and $156 for suburban, a difference of 66 percent. The entire suburban finding — the one that motivates the December 2 meeting, that Chapter 5 decomposed and Chapter 7 modeled — vanishes at the wrong grain and reappears at the right one, and nothing in the software distinguishes the two cases. There is no error message, no warning, and no visual difference between the correct chart and the incorrect one. The only defense is the one Section 3.3 installed: state the grain the metric requires before building the view, and check that the view's dimensions and aggregation produce it.

The practical shape of that defense in Tableau is worth stating as a routine, because it is short. Before building any view, write down what one mark is supposed to represent — one store, one month, one customer, one order. Then look at what the view's dimensions actually make one mark represent. If the two differ, either change the dimensions or move the aggregation upstream into Tableau Prep, which is what Lab 12.1 does and which is generally the better answer for a metric that will be used repeatedly. Calculated fields can express the correction inside the workbook, as the average-order-value expression above does, and this chapter uses them only at that introductory level; the row-level-versus-aggregate distinction that governs when such expressions succeed, and the level-of-detail expressions that solve the general version of the grain problem, belong to the advanced visual-analytics work that follows this guide.

12.11 From Prep to First Charts

The concepts are in place and the tool's vocabulary is mapped. This section describes the path from the certified file to the first views — what Tableau Prep is for, what belongs in it rather than in the workbook, how the core chart forms are constructed, and what faceting buys — and it is deliberately short on click paths, because the click paths belong in the labs of Section 12.13 where a student can follow them at the keyboard.

Tableau Prep's role in this course is narrow and specific: it is where the data is brought to the grain and shape the analysis requires, and where the decisions that would otherwise be repeated in every workbook are made once. `transactions_enriched` arrives already denormalized — the joins to `customers`, `products`, and `stores` that Chapter 4 taught have been performed, which is precisely what makes it a Tableau file rather than a modeling file — so Prep is not being used here for joining. It is being used for three things. First, inspection: Prep's profile pane displays, for every field, the distribution of values and the count of nulls, which is Chapter 4's inspection routine rendered as an interface and which should be read the same way, looking for the categorical variants, the impossible dates, and the nulls that the defect catalog taught students to expect. Second, derivation:

fields the analysis needs repeatedly — a month or week truncation of `order_date` with the week start declared explicitly, a store wave taken from the store roster rather than grouped by hand, a store-age-in-weeks field that Lab 12.1 Part C will need — belong here rather than in each workbook, because a definition that lives in one place can be corrected in one place. Each of those derivations carries its own inspection: check for negative store ages, for stores with a missing `opening_date`, for transactions dated before their store opened, and for how the online location is treated, since `ONLINE` has no opening date and therefore no store age at all. Third, and most importantly, aggregation to a stated grain, which is the answer to Section 12.10's trap: an aggregate step that rolls order lines up to orders produces a file on which average order value is simply an average, and on which no future analyst can compute it wrongly.

That last move deserves a caution, because it is a real trade-off rather than a free improvement. Aggregating to order grain discards `product_id`, `category`, and `is_occasionwear`, which are line-level facts, so the composition question of Section 12.5 cannot be answered on the aggregated file. The professional answer is not to choose but to output both: a line-grain extract for composition and product questions, and an order-grain extract for order-level metrics, with a documented reconciliation between them. The reconciliation is the verification, and it is exactly one number — total revenue must agree to the cent across the two outputs, because aggregation redistributes rows and must not create or destroy dollars.

Construction of the core forms in the workbook then follows the grammar directly, and the mapping is worth stating once so that the labs can move quickly. A field on the Columns shelf and a field on Rows establish the positional encodings; the Marks card sets the mark type and receives the non-positional encodings, with fields dropped onto Color, Size, Label, Detail, and Tooltip; and the aggregation of any measure is set on the field itself. A sorted bar chart is a dimension on Rows, a measure on Columns, and a sort applied to the measure. A trend line is a continuous date on Columns, a measure on Rows, and a Line mark. A scatterplot is two measures on the two shelves with the unit of analysis on Detail — and that last element is the one beginners omit, which is why their scatterplot shows one point: without a dimension on Detail, Tableau aggregates everything into a single mark, which is Section 12.10's lesson arriving in its most visible form.

The last construction the chapter owns is the one that will do the most analytical work in Lab 12.1.

DEFINITION

Small Multiples and Faceting

Small multiples are a series of small charts sharing identical axes, scales, and encodings, each showing the same measures for a different subset of the data, arranged in a grid so that the subsets can be compared by scanning. Faceting is the operation that produces them: partitioning the data by a dimension and drawing one panel per value of that dimension. Because every panel shares a scale, differences between panels are read as differences in the data rather than as differences in the drawing; because the panels are separate, the comparison is made across identical non-aligned scales rather than against a common baseline, which costs some precision in exchange for the ability to compare many subsets at once.

Source: Adapted from Tufte (2001) and Munzner (2014).

In other words, faceting trades one rung of the perceptual hierarchy for the ability to see seventeen things at the same time, and in analysis that trade is very often correct. StyleCraft's case makes the argument by itself. Weekly revenue for all seventeen stores on one set of axes is a thicket in which nothing can be traced; the same data as seventeen panels sorted by opening date is a picture in which each store's ramp shape can be traced, and in which the apparent underperformance of the newest stores can be tested against the possibility that they are simply four months old. Putting that question where it can be answered is the analytical payoff of the whole chapter, it is what the December 2 meeting most needs to have seen before it starts, and no aggregate chart produces it — which is the

general case for small multiples, and the reason they are the last thing this chapter builds before turning the tool over to the assistant.

12.12 AI as a Chart-Drafting Assistant

The division of labor this guide has refined since Chapter 7 — the assistant drafts mechanics, the analyst audits meaning — meets a new complication in Part III, and the complication is that a chart's mechanics and its meaning are produced by the same act. In Chapter 8 an assistant could write a model-fitting cell whose correctness was separable from the question of whether the model answered anything; the code either ran or did not, and the meaning was audited afterward against outputs the analyst could read. A chart has no such separation. The specification that says *bar mark, revenue on length, quarter on the category axis, vertical range 240,000 to 255,000* is simultaneously the mechanics and the claim, and the assistant that produced it made a claim about what the reader should conclude in the same breath as it chose a mark type. There is no later stage at which the meaning gets audited unless the analyst inserts one, which is what this section installs.

The chapter's verification theme is that insertion, and it is unusually cheap to run. The assistant returns a chart specification or a set of build instructions; the analyst reads the specification, predicts against Sections 12.3 through 12.9 what the resulting picture will show and where it will mislead, *then* builds it and grades the prediction. Predicting first is what makes the audit possible, because a chart that has already been rendered is persuasive in a way its specification is not — it looks finished, it looks designed, and the human tendency is to evaluate whether it is attractive rather than whether it is right.

The failure modes are specific enough to be listed for the audit, and they follow directly from what an assistant is and is not. The truncated axis to order is the first and most consequential: an assistant asked to "make the difference clear" or "show the growth" will do exactly that, and the most reliable way to make a 4.6 percent difference clear is to restrict the axis, which the assistant will do without comment because the instruction was followed. The grain slip is the second and the costliest, and it is Section 11.11's failure arriving in visual form: asked for "average order value by metro" against a line-grain file, an assistant will very often return an average of `line_revenue`, which is a correct average of the wrong thing, and the resulting chart shows a 3 percent gap where a 66 percent gap exists. The default pie is the third: composition is the one purpose for which the assistant's training data is saturated with a bad answer, and a seven-category pie will be proposed for StyleCraft's product mix unless the prompt forbids it. The palette mismatch is the fourth, in both directions — an ordered field such as `loyalty_tier` given a categorical palette that discards the order, or an unordered field such as `home_metro` given a sequential ramp that invents one — and it is accompanied by the accessibility omission, since nothing in the request will have mentioned color-vision deficiency and nothing in the assistant's default behavior will raise it. The invented field is the fifth and the easiest to catch: assistants routinely reference columns that would be natural for the question and do not exist in the file, an `aov` or a `customer_ltv` or a `region` where the schema has `home_region`, and the failure surfaces as an error rather than as a wrong chart only if the analyst runs the build rather than trusting it. The wrong-purpose chart is the sixth: asked whether suburban stores outperform urban ones, an assistant will frequently return a time series, which is a trend answer to a comparison question and which will look entirely reasonable. The undisclosed filter is the seventh and the most dangerous in a pre-read, because it leaves no trace in the image: nulls dropped, `ONLINE` excluded as "not a store," a date range quietly restricted to complete months. The default sort is the eighth and the most trivially fixable, and it is left in place constantly. And the overclaiming title is the ninth: asked to title a chart, an assistant produces a fluent sentence containing a verb the view has not earned, and the verbs are the ones Section 11.10's ladder governs — *drives*, *causes*, *proves* — attached to a description that licenses none of them.

Where the assistant genuinely helps, the list is long, and it is longer here than in any previous chapter, because more of visualization's labor is mechanical than practitioners like to admit. Enumerating candidate forms for a stated purpose is a good use: asked for five ways to display the composition question with their trade-offs, an assistant produces a better slate than most analysts generate unaided, and the analyst's judgment is then spent on choosing rather than on remembering. Prep step logic is boilerplate, as are calculated-field expressions of the kind Section 12.10 used, and the assistant's syntax is generally correct where its grain judgment is not — which is a useful

asymmetry to exploit deliberately, by specifying the grain in the prompt and delegating only the expression. Generating a color-blind-safe categorical palette as an explicit list of hex values, with the grayscale check described, is a task the assistant does quickly and well. Drafting the footer discipline of Section 12.9 — source, window, filters, grain, refresh date — is pure formatting. Writing alternative text for a chart, which almost nobody does and which accessibility increasingly requires, is a natural fit. And the adversarial use is once again where the assistant earns most: handed a description of a finished view and asked to list every way it could mislead a reader, it produces a checklist that will contain items the builder missed, at a cost of thirty seconds.

The governing instrument extends the audit lineage one more time. Table 8.5's five points do not all transfer, and saying which is part of the audit: *frame* becomes the question the view was built to answer, *baseline* becomes the reference the reader is meant to read the values against, and *error in decision units* becomes whether the view's quantities are in units a decision-maker can act on, while *leakage* and *split hygiene* have no visual analogue and are marked not applicable rather than skipped silently. Table 12.5 adds the three checks visualization makes necessary, and it is ordered so that the cheapest and most frequently violated check runs first.

Table 12.5

The Visualization Supplement to the Five-Point Audit (run with Table 8.5)

Supplemental point	The check	Fails when
V1. Grain and aggregation	State what one mark represents before looking at the view, then confirm the view's dimensions and the measure's aggregation produce exactly that; verify any per-order, per-customer, or per-store metric against a hand computation on one group	A per-order metric is computed from line-grain rows; a scatterplot shows one mark because no dimension is on Detail; totals change when an unrelated dimension is added
V2. Scale and baseline honesty	Read the axis floor on every length-encoded mark and confirm it is zero; read the range on every position-encoded axis and confirm the range was chosen rather than inherited; confirm any dual axis is justified or replaced; confirm the displayed window covers at least one seasonal cycle and that its start is defensible	A bar axis begins at a non-zero round number; two series track implausibly well on twin axes; a window begins at an event with no pre-period shown
V3. Encoding fits the purpose	Name the comparison the reader must make, name the channel carrying it, and confirm the channel's rank in Section 12.4 is appropriate; confirm palette type matches the field's level of measurement and survives a grayscale check; confirm the sort order was chosen; confirm the reference the reader needs is on the chart	Magnitudes are carried by angle or area; a categorical field is in a sequential ramp; bars are sorted alphabetically; a value is displayed with nothing to read it against

AI IN PRACTICE

Predict the Faults Before You Look

The chapter's verification theme, run as a five-step routine that takes about ten minutes and should be run on every AI-drafted view before it is shown to anyone. Step one is the request, written to expose rather than to conceal the assistant's assumptions: "I have `transactions_enriched` at order-line grain, with these columns [paste the schema]. I want to answer this question: do suburban and resort stores convert first-time buyers into repeat buyers at a lower rate than urban stores? Propose three candidate views. For each, state the mark type, every field-to-channel encoding, the aggregation and the grain one mark represents, the axis ranges, the sort order, and the palette. Do not use any column not in the schema I pasted; if the question requires a field I have not given you, say so instead of assuming it. Do not render or describe conclusions." The final two clauses do most of the work: the first defeats the invented field, the second stops the assistant from telling you what you are about to see. Step two is the prediction, made from the specification alone and written down before anything is built: for each candidate, name the comparison the reader will make, the channel it will be made through, the rank of that channel in Section 12.4, and the two faults you expect — then commit to which of the three candidates you expect to win and why. Step three is the build, in Tableau, following the specification exactly as written, including the faults; a specification corrected silently during construction cannot be audited. Step four is the audit and the grade: run the three transferable points of Table 8.5 and Table 12.5's three, line by line, on the built view, then compare the faults you found against the faults you predicted, and record both the misses and the false alarms, because a fault you predicted and did not find is as informative about your own model of the assistant as one you missed. Step five is the repair against Appendix E, one mechanic at a time, with a one-sentence note per repair recording what changed in the reading — not what changed in the chart. File both exchanges per Appendix D. And observe the rule that gives the routine its teeth: the assistant may propose any view it likes, but the analyst who circulates it owns every encoding in it, which is the visual form of signing the recommendation as the analyst of record.

12.13 Hands-On Application in Tableau

The preceding sections built the vocabulary; this section applies it to StyleCraft's own file in two labs that follow the pre-read's chronology rather than the chapter's section order. Lab 12.1 constructs: a Tableau Prep flow that certifies `transactions_enriched` at the two grains the analysis needs, six views built from the six analytic purposes of Section 12.5, and a faceted view that resolves the newest stores' apparent underperformance into a fact about store age. Lab 12.2 corrects: the AI round trip of Section 12.12, run on the deck circulating since Thursday, with each mechanic diagnosed against Table 12.3 and repaired against Appendix E.

One convention differs from Chapters 4 through 11 and is worth stating before the first box. Those chapters' labs presented runnable code cells; a Tableau lab's equivalent is a build path, so the boxed element here is labeled **Build** and contains the sequence of actions that produces the view, followed by the same plain-language account of input, transformation, and output that a code cell receives. Appendix A covers Colab mechanics and does not cover Tableau; students who have not opened Tableau Desktop or Tableau Prep before should work through the vendor's own Getting Started tour before this lab, and the Prep and Desktop versions assumed are those specified in the course technology requirements. AI assistants may draft any build path in this section, and every output is predicted before it is produced.

The labs use the certified `transactions_enriched` file, which is the denormalized join of `transactions` to `customers`, `products`, and `stores` described in the course data dictionary: one row

per order line, carrying `order_id`, `order_line_id`, `customer_id`, `product_id`, `store_id`, `channel`, `order_date`, `quantity`, `unit_price`, `discount_pct`, `line_revenue`, `line_cost`, and `campaign_id` from the fact table, the customer attributes `signup_date`, `acquisition_channel`, `home_metro`, `home_region`, `age_band`, `generation`, `gender`, `email_opt_in`, `sms_opt_in`, `app_user`, and `loyalty_tier`, the product attributes `category`, `subcategory`, `launch_date`, and `is_occasionwear`, and the store attributes `store_name`, `city`, `state`, `metro`, `store_type`, `opening_date`, `latitude`, and `longitude`.

12.13.1 Lab 12.1, Part A: The Prep Flow and the Two Grains

Everything in the two labs depends on a fact established in Section 12.10 and easy to forget under deadline: `transactions_enriched` is at order-line grain, and half the questions the pre-read must answer are at order grain. This part builds the flow that produces both, and it verifies the flow with a single number.

BUILD

The certified flow in Tableau Prep

1. Open Tableau Prep Builder. **Connect to Data → Text file →** select `transactions_enriched.csv`. The input step appears; in the input pane, confirm the field types, and set `order_date`, `signup_date`, `opening_date`, and `launch_date` to Date.

2. Click the **+** on the input step **→ Clean Step**. In the profile pane, read every field: the value distribution, the number of distinct values, and the null count. Record the row count shown in the step header.

3. Branch the flow *before* any date filter is applied, because one derived field must be computed from the full history. Click **+** on the clean step **→ Aggregate**. Drag `customer_id` to **Grouped Fields** and `order_date` to **Aggregated Fields**, set to Minimum, and rename the result `First Order Date`. This branch reads every row in the file, including the partial months trimmed below and any purchase that precedes the displayed analysis window. Confirm that the step returns exactly one row per customer.

4. Return to the clean step and take the second branch, which is the analysis window. Right-click `order_date` **→ Filter → Range of Dates**, and set the range to begin on the first day of the first complete month and end on the last day of the last complete month in the file. Record the new row count and the number of rows removed. The partial months at each end are trimmed because a monthly view built on them shows two artificial dips that are properties of the extract rather than of the business.

5. Still on the filtered branch, create the derived fields the analysis needs repeatedly. **Create Calculated Field →** name it `Order Month`, formula `DATETRUNC('month', [order_date])`. Create `Order Week` with `DATETRUNC('week', [order_date], 'monday')`. Create `Store Age Weeks` with `DATEDIFF('week', [opening_date], [order_date], 'monday')`. The week-start argument is not optional in this course: omitted, Tableau uses the data source's configured start of week, and two students on differently configured sources will produce different weekly bins and different store ages from the same file (Tableau, 2026c, 2026h). Both `DATETRUNC` and `DATEDIFF` accept the optional `start_of_week` argument in Prep as well as in Desktop.

6. Create `Store Wave` from the canonical `store_wave` field on the store roster. If your roster does not carry one, create it as an explicit calculated field that lists the store identifiers assigned to each wave — `IF [store_id] IN ('SOHO','FLAT','BKLN') THEN 1 ELSEIF … END` — rather than as a manual grouping on `opening_date`. A manual grouping cannot be audited from a screenshot and is easily reconstructed wrongly by the next person to open the flow.

7. Join the full-history branch to the filtered branch. Click **+** on the filtered clean step **→ Join**; set the join type to **Left**, with the filtered line-level data on the left and the `First Order Date` table on the right, joined on `customer_id`. Read the join profile before continuing: the left row count must be unchanged, every filtered transaction must have matched exactly one customer, and no row may show a `First Order Date` later than its own `order_date`. Prep's aggregate step changes grain according to the fields you group by, and its join profile reports row inclusion for exactly this kind of check (Tableau, 2026b).

8. On the joined step create `Is Repeat Order` with `IF [order_date] > [First Order Date] THEN 1 ELSE 0 END`. Computing it here rather than in the workbook is the general principle of this step: a field whose correct value depends on a grain other than the view's belongs upstream. Note what the rule does with same-day orders. Because `order_date` carries a date rather than a timestamp, every order placed on a customer's first purchase date is classified as a first-day order rather than as a repeat. That is a defensible convention and it is declared here; it is not the same thing as identifying the first individual order, and any statement built on this field should say so.

9. Click the + on the joined step → **Output**. Name it `lines_certified`, set the output type to **Tableau Hyper Extract**, and note the path. This is the line-grain file.

10. Return to the joined step, click + → **Aggregate**. Drag to **Grouped Fields**: `order_id`, `customer_id`, `order_date`, `Order Month`, `Order Week`, `Is Repeat Order`, `store_id`, `store_name`, `city`, `state`, `latitude`, `longitude`, `metro`, `store_type`, `channel`, `home_metro`, `generation`, `loyalty_tier`, `app_user`, `Store Age Weeks`, `Store Wave`, `First Order Date`. Drag to **Aggregated Fields**: `line_revenue` (Sum), `line_cost` (Sum), `quantity` (Sum), and `order_line_id` (Count, renamed `Lines In Order`).

11. Click + on the aggregate step → **Output**. Name it `orders_certified`, again as a Hyper extract.

12. **Run Flow**. Record the row count of each output.

The input step reads the flat file; the first aggregate step computes each customer's first order date from the full history, before any window is imposed; the clean step performs the window filter and the derivations that would otherwise be repeated in every workbook; the join attaches the first-order date without changing the line count; and the second aggregate step collapses order lines into orders by grouping on every field that is constant within an order and summing the fields that are not. The order of those steps is the substantive lesson of the flow. Computing a customer's first purchase after trimming partial months would classify as a first purchase any order that merely happens to be the earliest one still visible, which misclassifies every customer whose true first purchase fell in a trimmed month or before the analysis window opens. The output is two certified extracts describing the same business at two grains — one on which product and category questions are answerable, one on which order-level metrics are simply averages. Note what the grouped-field list encodes: every field placed there is an assertion that the field does not vary within an order, and an assertion that turns out to be false will silently split one order into several rows, which is why the verification below counts orders as well as dollars. The store coordinates and the customer's first-order date are carried through the grouping for the same reason — the map of View F and the repeat-order metric of View B both need them at order grain, and a field that the workbook would have to reconstruct is a field the workbook can reconstruct wrongly.

VERIFICATION CHECK

Predict every number below before running the flow, then reconcile rather than spot-check. The row counts and totals quoted here — 97,912 input lines, 96,844 after the complete-months filter, 45,206 orders, and $4,882,247.60 of revenue — are the values the certified StyleCraft extract is built to return; confirm them against the answer key published in the companion repository before treating any of them as a target and report a mismatch as a finding rather than adjusting the flow until the numbers appear. The line-to-order ratio should come to 2.14 — predict it first from what Chapter 5 established about basket size and note that a ratio near 1.0 would mean the grouping keys split orders and a ratio far above 3 would mean orders were merged. Revenue reconciliation is necessary and not sufficient. Confirm all of the following across the two outputs: the row count of `orders_certified` equals `COUNTD(order_id)` in `lines_certified`; total revenue, total cost, and total quantity agree to the cent or to the unit; distinct customers and distinct stores agree; and the minimum and maximum `order_date` agree. Confirm separately that the join in step 7 preserved the line count exactly. Total revenue must agree to the cent because an aggregation redistributes rows and must never create or destroy dollars. Finally, compute average order value two ways on the two files — `SUM([line_revenue]) / COUNTD([order_id])` on the line-grain extract, and a simple `AVG([line_revenue])` on the order-grain extract — and confirm they return the same $108.00, which is the chain average this chapter's reference lines will use. Then compute `AVG([line_revenue])` on the *line-grain* file, observe that it returns $50.41, and write down what a reader would conclude if that number were labeled "average order value." That single wrong label is the most common error in this chapter and the hardest to see, because the chart it produces is beautiful.

12.13.2 Lab 12.1, Part B: Six Purposes, Six Views

With both grains certified, the pre-read can be built. The discipline of this part is that each view is specified before it is constructed — purpose, comparison, encoding, grain, reference — and that the specification is written down where it can be graded. Table 12.6 is that specification for all six, and filling it in is the assignment; the build paths below construct three of the six, and the remaining three are the student's.

Table 12.6

The Expansion Story in Six Views

View	Purpose (Section 12.5)	The comparison the reader must make	Grain of one mark	Reference the view must carry
A. Order value distribution by metro	Distribution	Are suburban orders uniformly larger, or larger and more dispersed?	One order	The chain median, drawn
B. Repeat-order share by store type	Comparison	In which store types is the largest share of orders placed by customers who had already bought before?	One store type	The chain share for the same store population, drawn
C. Weekly revenue with the openings marked	Trend	Does the series step at the Greenwich opening, and does it step again later?	One week	Dated vertical marks at each opening; one full seasonal cycle visible

View	Purpose (Section 12.5)	The comparison the reader must make	Grain of one mark	Reference the view must carry
D. Items per order against occasionwear share, by store	Relationship	Do the stores with big baskets also sell the occasionwear mix, or are these two different stories?	One store	The chain averages on both axes, as crossed reference lines
E. Category revenue share by metro	Composition	Does the suburban category mix differ from the urban one, and by how much?	One category within one metro	None required; shares are self-referencing
F. Store locations with trailing revenue	Geographic	Where are the newest stores, and does geography explain the ramp?	One store	A labeled size legend with two reference sizes

BUILD

View B — repeat-order share by store type (comparison)

1. Connect a new workbook to `orders_certified`.

2. Confirm that `Is Repeat Order` arrived from the Prep flow of Part A. It is already correct at order grain, which is why it was built there.

3. Create `Repeat-Order Share` as `SUM([Is Repeat Order]) / COUNTD([order_id])`. Format it as a percentage with one decimal. Name it precisely, because the obvious name is wrong: this is the share of *orders* placed after the customer's first purchase date, subject to the same-day convention declared in Part A, and it is not the share of customers who returned. A view labeled "returning-customer share" invites a customer-level reading the metric does not support.

4. Drag `store_type` to **Rows** and `Repeat-Order Share` to **Columns**. The Marks card should read Bar; if it does not, set it. Exclude `store_id = ONLINE` on the Filters shelf, since the comparison is across physical store types.

5. Click the sort icon on the `Repeat-Order Share` axis to sort descending. Confirm the sort is applied and not merely apparent.

6. Right-click the horizontal axis → **Edit Axis** → confirm **Include zero** is checked and the range starts at 0.

7. Supply the chain reference deliberately, because a constant line requires a value and does not compute the pooled rate for you. Tableau's reference lines can be based on a constant or on a computed value, but the computation and its scope must be specified (Tableau, 2026d). Choose one of three routes and record which you used: compute the verified chain-wide share separately and enter it as a constant; precompute it in Prep and carry it into the extract; or create a parameter populated from the certified answer key. **Analytics** pane → drag **Constant Line** to the view → enter that value → label it "Chain average (physical stores)."

7a. Do not use the average of the four displayed store-type rates. The chain rate is computed on the pooled denominator and differs from the unweighted mean of the four rates, for the reason Chapter 5's denominator discussion gives. And because step 4 filters `ONLINE` out, the reference must be the pooled rate for the same physical-store population; if a comparison against the whole business including online is intended, draw a second, separately labeled line rather than redefining the first.

8. Add the footer: **Worksheet → Show Title**, and in the caption record the source extract, the date window, the filters, the grain ("one bar = one store type; share = orders placed after the customer's first purchase date ÷ distinct orders"), and the refresh date.

Input is the order-grain extract; the transformation is a ratio of two aggregations computed within each store type; the output is a sorted bar chart with a reference line. Three of the eight steps are encoding discipline rather than construction — the sort, the zero baseline, and the reference line — and they are the three that a five-minute version of this chart omits.

BUILD

View C — weekly revenue with the openings marked (trend)

1. New worksheet on `orders_certified`. Drag `Order Week` to **Columns**; right-click the pill and confirm it is **continuous** (green), not discrete.

2. Drag `line_revenue` to **Rows** with aggregation Sum. Set the Marks card to Line.

3. Drag `metro` to **Color**. State the emphasis hierarchy before choosing hues: NYC Suburban is the primary subject of the December 2 question and Resort is secondary. Assign neutral gray to the two metros not under examination, one saturated accent hue to NYC Suburban, and a muted variant of that accent — same hue, lower saturation — to Resort, so the pairing reads as primary and secondary rather than as two competing signals.

4. **Analytics** pane → **Reference Line** → Table → set Value to a constant at each opening date, or add an annotation: right-click the point on the axis at the Greenwich opening date → **Annotate** → **Point**, and label it "Greenwich opens."

5. Repeat the annotation for the Wave 4 opening dates, keeping the labels short.

6. Right-click the vertical axis → **Edit Axis**. Because the marks are lines and encode value by position, the axis range may be restricted; set it deliberately and record the choice in the caption rather than accepting the default.

7. Confirm the horizontal range spans the full twenty-four months, so that at least one complete seasonal cycle is visible per Section 10.3.

8. Add the footer as in View B.

Input is the order-grain extract; the transformation is a sum of revenue within week and metro; the output is a multi-series line chart with dated annotations. The color choice in step 3 is the emphasis discipline of Section 12.8 rather than a decoration: four saturated hues would make the chart about the existence of four metros, while two neutral grays plus one accent and its muted variant make it about the suburban series first and the resort series second, which is the hierarchy the question implies.

BUILD

View F — store locations with trailing revenue (geographic)

1. New worksheet on `orders_certified`. Confirm `latitude` and `longitude` carry geographic roles; if not, right-click each → **Geographic Role** → Latitude / Longitude. Either use Tableau's generated Latitude and Longitude fields or use the store roster's own coordinates.

2. Place the coordinates as continuous measures, which is how Tableau treats latitude and longitude in the standard proportional-symbol workflow and where its generated fields appear (Tableau, 2026f, 2026g): `AVG(Longitude)` on **Columns** and `AVG(Latitude)` on **Rows**. Drag `store_name` to **Detail** so that the average is taken within one store and one mark is one store.

3. Set the Marks card to Circle. Drag `line_revenue` to **Size**. Filter to the trailing twelve months using the anchored window defined below rather than a relative-date filter.

4. Open the quantitative **Size** legend → **Edit Sizes** → set **Sizes vary** to **From zero**, which is the mapping that makes circle area proportional to the value from an origin of zero (Tableau, 2026f). Retain the continuous size legend, and add a written note on the view stating what circle area represents. If the pre-read requires two fixed labeled reference circles, build them as a small separate legend worksheet; a standard quantitative size legend does not reliably provide two custom reference symbols.

5. Drag `Store Wave` to **Color**. Waves 1 through 4 are ordered, so a categorical palette would contradict Table 12.4. Use a sequential palette running from earlier to later waves, or — better for this question, which is about the newest stores rather than about the distinction among all four waves — neutral gray for Waves 1 through 3 and one accent hue for Wave 4. Then run the three-part accessibility check of Section 12.8.

6. Do *not* attempt to read magnitudes from this view. Place View B's bar chart beside it in the pre-read, per Section 12.7's division of labor.

Input is the order-grain extract with a trailing-twelve-month filter; the transformation is a sum of revenue per store; the output is a symbol map.

The trailing window is anchored to the dataset rather than to the clock, and this is a course-wide convention rather than a preference. A relative-date filter evaluates against the current system date, so a workbook built this term will silently return a different window next term and eventually no records at all from a static teaching extract. Define the end of the window as the maximum complete date in the certified extract and derive the start from it:

```
Analysis End Date   = #2026-06-30#

Trailing Start Date = DATEADD('day', 1,

                        DATEADD('year', -1, [Analysis End Date]))
```

Two details in that expression are the point of it. The date is written as a Tableau date literal between hash marks, or supplied as a Date parameter, because a bare `2026-06-30` is not valid syntax in a calculated field. And the start is derived by stepping back one year and forward one day rather than back twelve months, because an inclusive filter anchored twelve months back would begin on June 30, 2025 and return twelve months plus a day. The window is therefore July 1, 2025 through June 30, 2026, and it remains reproducible in 2027 and beyond, so that any student who reruns the workbook gets the figures the answer key documents. Step 6 is the substantive

instruction: the map's job is to answer whether location explains the ramp, and the honest answer will come from the faceted view of Part C rather than from the circles.

> VERIFICATION CHECK
>
> Predict each view's shape before building it, in one sentence apiece, and predict the specific numbers where Part II already supplies them. View A should show suburban orders both higher and substantially wider than urban ones, with the suburban distribution's right tail carrying the occasionwear baskets; predict the two medians before you see them. View B's repeat-order share should return roughly 61 percent for Urban, 33 percent for Flagship, 24 percent for Suburban, and 19 percent for Resort — predict the ordering first, from what Chapter 5 and Chapter 9 established. Then predict the chain reference line, and notice that it is not the average of those four numbers: the chain-wide share sits near 47 percent because urban stores carry most of the chain's orders, well above the unweighted mean of 34 percent. A student who predicts 34 and finds 47 has not found an error in the extract; she has found the difference between an average of rates and a rate computed on the pooled denominator, which Chapter 5's denominator discussion already named. View C should show the Greenwich step Chapter 10 already found, and it should show it as a modest step on a rising, strongly seasonal series rather than as a takeoff; if the step looks dramatic, check the axis range you set in step 6 before believing it. View D should show the two axes are correlated but not redundant, with at least two stores off the diagonal, and those stores are worth naming in the pre-read. View E should show occasionwear's share roughly three times higher in suburban than in urban metros while average line value differs by about 3 percent, which is the designed finding of Chapter 5 restated visually and the single most important slide in the packet. Predict that View F will show no geographic pattern that survives inspection — and hold the prediction as a prediction. If the certified data does show a spatial pattern, that is a result to investigate against store age and store type, not a defect in the view. Then run V1 from Table 12.5 across all six: for each view, state what one mark represents, and confirm it against the dimensions actually in the view.

12.13.3 Lab 12.1, Part C: Small Multiples and the Store-Age Artifact

The pre-read's hardest question is the one the real-estate director will ask first: the newest stores look weak, so why open eight more? Every aggregate view in Part B says the Wave 4 stores underperform, and every aggregate view is telling the truth about a comparison that should never have been made. This part builds the view that can settle it, and it is the chapter's demonstration that faceting is an analytical instrument rather than a layout preference.

BUILD

Seventeen panels, one scale

1. New worksheet on `orders_certified`. Drag `Store Age Weeks` to **Columns** (continuous) and `line_revenue` to **Rows** (Sum). Set the Marks card to Line.

2. Drag `store_name` to **Rows**, to the left of the measure. Tableau now draws one row of panels per store.

3. Right-click `store_name` → **Sort** → **Field**, sort by `MIN(opening_date)` ascending, so the panels run oldest to newest. Sort by the field rather than manually, so that the panel order is reproducible by anyone who opens the workbook.

4. **Analysis** menu → **Table Layout** → confirm the axes are *not* independent: every panel must share one vertical scale, or the comparison the small multiple exists to support is destroyed.

5. Set the panel size small enough that all seventeen are visible without scrolling; use **Fit** → **Entire View** and reduce the row height.

5a. Before the benchmark can be computed, build a complete store-week scaffold, because the order-grain extract contains no row for a week in which a store took no orders. Generate one row per physical store for every store-age week from 0 through 25, left-join weekly revenue onto it, and set revenue to zero for any store-week in which the store was open and observable but recorded no orders. Leave a store-week null — rather than zero — where the store was not yet open or the window had not yet closed, so that an absent observation is never read as a zero.

5b. Carry **both** horizontal variables on the scaffold, because the two panels of this lab differ only in which one is placed on Columns. `Store Age Weeks` is the scaffold's own key; derive the matching calendar week from it:

```
Order Week = DATEADD('week', [Store Age Weeks],

DATETRUNC('week', [opening_date], 'monday'))
```

Both displays are then built from exactly the same scaffold rows — store-age weeks 0 through 25 for every physical store — with only the field on Columns changing. That is not a convenience. The exhibit's whole claim is that nothing changes except the horizontal encoding, and it cannot be made honestly if the calendar-time panel silently draws on the full transaction history while the store-age panel draws on the first twenty-six weeks.

6. Add a reference line at the chain-average weekly revenue for stores in their first twenty-six weeks, so each new store's ramp is read against the ramp its predecessors had. Compute it on the scaffold, not on the view, and define it as

```
SUM(revenue for store-age weeks 0-25)

/ COUNT(scaffold store-weeks 0-25 that were open and observable)
```

so that every eligible store-week counts exactly once. Two failures are avoided here. Computing the average from the marks in the view would weight stores by how many order lines they generated rather than by the store-weeks the benchmark describes. And omitting the zero-order weeks — which is what the raw extract does silently — drops the weakest observations from the denominator and biases the benchmark upward, which is the direction that would make a new store's ramp look worse than it is.

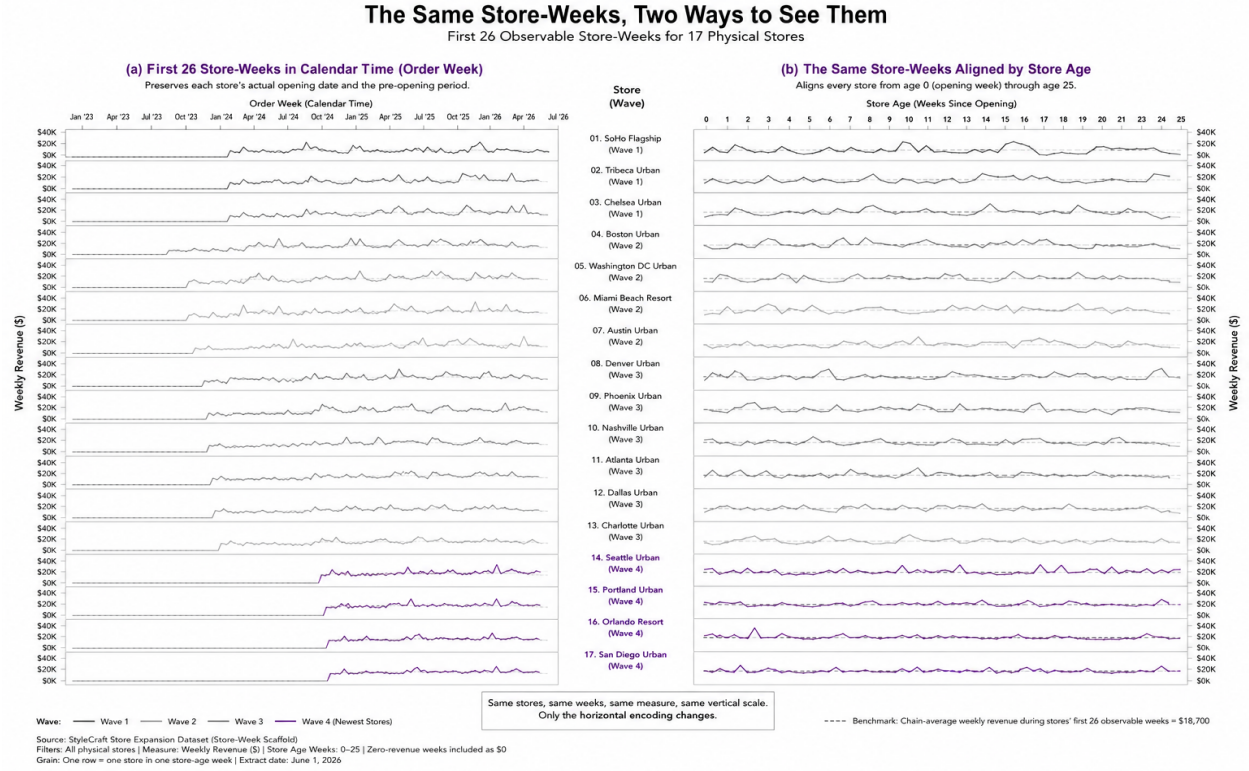
6a. Build the small multiples themselves on the same scaffold. Without it, Tableau draws a line straight across an absent week, so a store that sold nothing for a fortnight is rendered as though the fortnight never happened.

7. Duplicate the sheet, and on the copy replace `Store Age Weeks` with `Order Week` on Columns. Change nothing else — same scaffold, same rows, same measure, same vertical scale, same panel order, same styling.

Input is the order-grain extract; the transformation is a sum of revenue by store and by store age; the output is seventeen small charts on one scale. Step 7 is the crux of the lab, and Figure 12.6 is the result: the same data, faceted the same way, plotted against two different horizontal variables. On calendar time, the Wave 4 stores are eighteen months of white space followed by a short low line, and they look like failures. On store age, they are laid over their predecessors' first months, and the comparison becomes the one the question actually requires — whatever that comparison then turns out to show.

Figure 12.6

Calendar Time Versus Store Age



VERIFICATION CHECK

Predict the two pictures before building either or predict them precisely enough to be wrong. On the calendar-time facet, the Wave 4 panels will be empty for most of the horizontal range and will end with a short line at a level far below SoHo's; predict what a reader who saw only this panel would recommend. On the store-age facet, the designed expectation is that most Wave 4 ramps will fall inside the band traced by the older stores' first twenty-six weeks — predict which one or two will not, because the honest version of this finding is that store age explains most of the gap rather than all of it. Treat that expectation as something to verify rather than as the answer: a different result is a finding to investigate and report, not an error to edit away. Then run the comparison the whole part exists for: name, in one sentence, what changed between the two views, and confirm that nothing changed except the horizontal encoding. No filter was applied, no row was removed, no measure was redefined. The recommendation a reader would make is flipped because of a choice about what one axis means, which is the most important sentence in this chapter and the one Section 12.16 will hold the analyst responsible for. Finally, connect whatever you find to what the course data dictionary already documents about storage and marketing exposure — the confound Chapter 9 met as a fairness problem in a model, and which this view meets as an encoding problem in a chart. If the facet confirms the expectation, it is the same artifact caught by a different instrument; if it does not, you have found something the data dictionary does not yet describe, and that is the more valuable result.

12.13.4 Lab 12.2, Part A: The AI-Drafted Chart, Predicted and Diagnosed

The constructive lab is complete; the corrective one begins with the deck that has been circulating since Thursday. Run the five-step routine of Section 12.12's AI in Practice box end to end, documenting each move per Appendix D.

Begin with the request, and make it the *bad* request on purpose, because reproducing the failure is the point: ask an assistant to "make a chart showing how much the suburban stores have grown this quarter and make the growth clear." Do not paste the schema. Do not state the grain. Ask for a build path for Tableau.

Then stop and predict. Before reading what came back, write down five predictions: which mechanic from Table 12.3 the assistant will use to satisfy "make the growth clear"; whether it will compute a per-order or a per-line quantity; what it will do about the sort; whether it will name a column the file does not contain; and what verb its suggested title will use. Then read the response and grade the predictions.

BUILD

Reproducing the circulating chart, faults included

1. New worksheet on `lines_certified`. Drag `order_date` to **Columns** and set it to discrete **Quarter**. Filter to the two most recent quarters.

2. Filter `store_type` to Suburban and Resort.

3. Drag `line_revenue` to **Rows** with aggregation Sum. Marks card: Bar.

4. Right-click the vertical axis → **Edit Axis** → uncheck **Include zero** → set the range from 240,000 to 255,000.

5. Add a second sheet: `Order Month` on Columns, `line_revenue` (Sum) on Rows as a line, then drag a running count of open stores to Rows and right-click → **Dual Axis**; do not synchronize the axes.

6. Add a third sheet: `category` on Color, `line_revenue` on Angle, Marks card set to Pie.

7. Add a fourth sheet: `loyalty_tier` on Color with the red-to-green diverging palette, `line_revenue` on Rows as bars.

8. Do not repair anything yet. Screenshot all four.

Input is the line-grain extract; the transformation is a set of sums; the output is the deck as circulated, rebuilt so that its mechanics can be inspected from the inside rather than argued about from the image. Building the faults deliberately is what makes them legible: an analyst who has set an axis floor to 240,000 by hand will never again fail to notice one.

VERIFICATION CHECK

Diagnose all four against Table 12.3 and Table 12.5 before reading further and quantify each rather than merely naming it. For the first chart, compute the ratio of the two bars' *drawn heights* — with an axis from 240,000 to 255,000, the first bar occupies about 7 percent of the plot height and the second about 80 percent, a drawn ratio near twelve to one against a true value ratio of 1.046 to one — and then state the effect the way Section 12.7 states it, as a drawing that suggests roughly 1,100 percent growth where the data shows about 4.6 percent. Do not report a single exaggeration factor unless you also declare which two quantities you divided, because three different ratios are available here and they differ by more than an order of magnitude. For the second, change the right axis's range by two stores and observe that the coupling disappears; then state the second, more serious objection, which is the one from Section 7.5 and has nothing to do with drawing. For the third, cover the labels and try to rank the middle four categories; then rebuild the same data as a sorted bar and try again. For the fourth, export the image to grayscale and count how many of the four loyalty tiers remain distinguishable. Then run V1 across all four sheets and find the fifth fault the deck's readers have not noticed. Line grain is not itself an error: three of the four charts display sums of revenue, for which the line-grain extract is entirely valid. The fault is the mismatch between the extract and a claim made about it — the covering email calls the loyalty-tier chart "average order value," and the worksheet computes `AVG(line_revenue)`, which is average line value. State the rule that follows, because it is the one students most often get backward: a chart built on line-grain data is wrong only when the chart or the text around it makes a per-order, per-customer, or per-store claim that the grain does not support.

12.13.5 Lab 12.2, Part B: The Repair, and What Changed in the Reading

The last part rebuilds the four charts against the Appendix E checklist, and it is graded on the notes rather than on the charts.

BUILD

The four repairs

1. Chart one: re-point the sheet at `orders_certified`. Right-click the axis → **Edit Axis** → check **Include zero**. Observe that the two bars are now nearly identical. Then decide whether the finding survives its honest rendering: if the 4.6 percent change is genuinely the point, duplicate the sheet, extend the filter to all eight quarters, switch the Marks card to Line, and add a reference line at the prior-year same-quarter value.

2. Chart two: delete the dual axis. Build two sheets — suburban and resort revenue by month; cumulative store count by month — and place them in a vertical arrangement sharing the horizontal axis. Add urban revenue to the first sheet in gray as the comparison series.

3. Chart three: change the Marks card from Pie to Bar, move `category` from Color to Rows, move `line_revenue` from Angle to Columns, sort descending, and confirm the axis includes zero. Then duplicate and convert to a 100 percent stacked bar split by `metro`, which is the view that answers the composition question the pie was pretending to answer.

4. Chart four: replace the red-to-green palette with a single-hue sequential ramp ordered None → Insider → VIP → Backstage. Run the grayscale check. Add the footer discipline of Section 12.9 to all four sheets.

Input is the two certified extracts; the transformation is a sequence of encoding changes with no change to any underlying query; the output is four charts that say what the data says. Note that step 1 contains the chapter's most uncomfortable moment, and it should not be skipped: the honest chart shows almost nothing, and the analyst must then decide whether there is a finding at all. That decision is analysis, and it is what the truncated axis was substituting for.

VERIFICATION CHECK

For each repair, write one sentence recording what changed in the *reading* rather than what changed in the chart — "a reader now sees a quarter that moved slightly, rather than a quarter that appeared to jump twelvefold" is the form. Then write the note that actually gets sent. The merchandising analyst who built the original deck is a colleague under the same deadline who used the tool the way it presented itself, and the memo that lands well is the one that names the mechanic rather than the person, shows the before and after side by side, and states the general rule so that the next chart is built correctly rather than corrected later. Draft it in four sentences and confirm before sending that it contains no sentence beginning with "You should have." Finally, count what the whole lab cost: four repairs, roughly thirty minutes, against a deck that had already reached the director of real estate. That ratio — minutes of audit against the credibility of a December meeting — is the argument for making the audit routine, and it is the argument Section 12.15 will find running through every organization that has ever adopted a chart standard.

12.14 Marketing Interpretation and Managerial Insight

The labs produced two certified extracts, six views, a faceted resolution of the store-age question, and four repaired charts. The pre-read circulates Monday, and what circulates is not the charts, but the sentences readers will form while looking at them. This section translates, and as in every chapter of Part II it does so partly by exhibiting the

wrong readings, because analytical charts attract their own family of misinterpretation — one that differs from the misreadings of models and forecasts in a specific way. A wrong reading of a regression coefficient requires the reader to know what a coefficient is. A wrong reading of a chart requires nothing at all, arrives instantly, and is held with the confidence of something seen rather than something inferred.

The first wrong reading is the one this chapter's outline predicted, and it will be spoken by whoever opens the circulating deck first: "Suburban revenue is surging — look at the jump last quarter." The reader has decoded a length ratio of roughly twelve to one, which is what the chart drew, and has translated it into the word *surging*, which is what a twelve-to-one ratio would justify. The underlying movement is 4.6 percent, from $241,000 to $252,000, which on a quarterly series with StyleCraft's seasonality is inside the range the last eight quarters have wandered through without anyone using a verb. The correction is not a hedge and should not be delivered as one: suburban and resort revenue rose 4.6 percent quarter over quarter, which is within the ordinary quarter-to-quarter variation of the series and is smaller than the holiday effect Section 10.3 already decomposed, and the chart that made it look otherwise had an axis floor of $240,000. Two features of that sentence are worth copying. It gives the number and the comparison in the same breath, and it names the mechanic rather than merely asserting that the impression was wrong — because the reader who was misled will believe a correction that shows her how she was misled and will resist one that simply contradicts what she saw.

The second wrong reading is the expensive one, and it does not come from a broken chart at all. "The Wave 4 stores are underperforming; we should stop the expansion." Every aggregate view supports it. Trailing-twelve-month revenue by store puts the eight newest at the bottom; the calendar-time facet shows them as short low lines; the map shows small circles in Dallas, Scottsdale, and Washington. And the comparison being made — eight stores that have been open four months against nine stores that have been open one to two years — is one that no analyst would make in a table and that the chart makes irresistible, because the eye compares whatever is placed side by side on a shared scale and does not ask how old anything is. The correction is the store-age facet of Lab 12.1 Part C, and the way to deliver it in a meeting is to show both pictures in sequence and let the room test its own conclusion against the second one. The general lesson is worth naming because it will recur in every visual comparison anyone builds: a chart's implicit claim is that the things it places on a common scale are comparable, and that claim is made by the layout rather than by the data. An analyst who would never run a regression without asking what the units have in common should not build a bar chart without asking the same question.

The third wrong reading belongs to the map, and it is the most seductive because geography feels explanatory. "The West Coast stores are our growth engine — look at Beverly Hills and Scottsdale." Two errors are stacked. The first is Section 12.7's: the reader is ranking magnitudes by circle area, a channel the perceptual hierarchy places near the bottom, and a difference she reads as large may be a difference of thirty percent or of three hundred. The second is subtler and is the reason maps are so often the wrong instrument in retail analytics: a map of a per-store quantity mostly draws where the stores are, and StyleCraft's stores are where they are because of a real-estate strategy rather than because of customer demand. The map answers "where" and cannot answer "why," and every use of it to answer "why" is the ecological reasoning Section 3.3.1 warned about arriving in cartographic form. The correction is the division of labor Lab 12.1 built into the pre-read: the map establishes location and adjacency, and the sorted bar beside it carries every magnitude claim anyone intends to make.

The pre-read that survives all three has a fixed anatomy, and its ordering is an argument in the same way Chapter 11's deliverable ordering was. Page one is the reading guide: one paragraph naming the question each of the six views answers, the grain of one mark in each, and the window all six share, so that a reader who disagrees knows exactly what she is disagreeing with. Page two is the composition finding, because it is the one that changes minds — occasionwear share roughly three times higher in suburban metros, with average line value differing by about 3 percent, which together establish that StyleCraft's suburban advantage is basket construction rather than pricing power and therefore that the advantage is a merchandising fact rather than a market fact. Page three is the distribution and comparison pair: suburban orders larger and wider, suburban repeat-order share less than half the urban share, both against drawn chain averages, which is the tension the December meeting exists to resolve. Page four is the trend with the openings annotated and the full seasonal cycle visible, which prevents the cherry-picked reading before anyone constructs it. Page five is the pair of faceted views from Lab 12.1 Part C, presented together on one page rather than separated, with the calendar-time facet on the left and the store-age facet on the right and a

single line between them stating that nothing changed except the horizontal encoding; presenting the pair rather than the conclusion is what keeps the page an analytical exhibit, and the question of how a finding should be *ordered* for a reader who has ten minutes rather than a week belongs to Chapter 13. Page six is the map, captioned to say what it does not establish. And an appendix carries the charts that were built and rejected, with a line apiece on why, which is the visual counterpart of Chapter 11's exploratory section and serves the same function: it makes the analyst's search visible, so that nobody has to wonder whether these six views are the six that worked.

One translation deserves its own paragraph, because it is what makes this chapter's deliverable different in kind from Part II's. Every artifact Chapters 8 through 11 produced was valuable for the number inside it. This chapter's artifact is valuable for a decision it makes possible in the other direction — the same asymmetry Chapter 11 found in experiments, arriving here in a cheaper form. The pre-read's highest-value output is not the six views that shipped. It is the four charts that were repaired before they reached the director of real estate, and the one conclusion that reversed when an axis changed meaning. Managers judge visualization work by the picture that made something clear. Analysts should judge it by the picture that stopped something wrong, and should say so, because the second kind is invisible by construction and therefore never gets credited unless the analyst names it.

12.15 Business Analytics in Practice

This section turns from the fictional case to how visualization standards, governed data sources, and dashboard portfolios operate inside professional organizations — where the governance of charts has quietly become an infrastructure problem, where a single axis has cost real credibility, and where the enormous investment in self-service dashboards has produced a graveyard that most analytics leaders can describe from memory.

The first vignette is the rise of the visualization standard, which in most large firms arrived not as a design initiative but as damage control. It is a composite drawn from recurring organizational practice rather than a report on one named firm, and the quantities in it are illustrative of the pattern rather than measured. The pattern is consistent enough to be a genre. A company adopts a self-service BI platform; within eighteen months it has several thousand workbooks, several dozen definitions of revenue, and a weekly meeting in which two directors present contradictory charts of the same metric and neither can reconstruct the other's filters. The response, at firms from national retailers to global banks, has been to build two things at once. The first is a style guide — a documented chart standard specifying the corporate palette with its sequential and diverging ramps, the mandatory zero baseline on bar charts, the required footer carrying source, window, filters, and grain, and an explicit list of prohibited forms that almost always names the three-dimensional chart and the multi-slice pie. The second, and the more consequential, is the certified data source: a governed, versioned dataset published by a data team with its joins, grain, and metric definitions fixed, which analysts connect to instead of building their own extract from the warehouse. The certification does for a chart what the data dictionary of Section 3.8 does for a metric — it moves a definition out of individual judgment and into shared infrastructure — and it directly prevents the failure Lab 12.1 Part A was built around, because a certified source publishes its grain and a workbook built on it cannot quietly compute a per-order metric from line-grain rows. The practice lesson for a marketing analyst is about where the argument goes. In an organization with a standard, the argument about whether a bar chart may start at 4,000 happens once, in a document, among people thinking clearly and without a deadline; in an organization without one, it happens every week, in a meeting, about a specific chart whose author is present and defensive. The standard's value is not aesthetic consistency. It is that it converts a recurring interpersonal negotiation into a settled rule.

The second vignette is the cost of a single axis, and it is worth telling carefully because the mechanism is more interesting than the scandal. Truncated axes appear regularly in investor presentations, product-launch decks, and public-health graphics, and they are almost never fabrications: the underlying numbers are correct, the axis is labeled, and the presenter can point to both. What the research shows is that none of that helps. Pandey et al. (2015) presented readers with matched pairs of charts — identical data, one drawn from zero and one truncated — and measured substantial shifts in readers' judgments of how large the differences were. Correll et al. (2020) pushed the finding further, testing whether making the truncation obvious repaired the damage, and found that readers' magnitude judgments remained distorted even when the axis break was clearly signaled. The practical consequence, seen repeatedly when a truncated chart from a corporate deck is picked up and redrawn by a journalist or a

competitor, is that the organization loses a specific and hard-to-recover asset: the presumption that its charts can be read at face value. Once an audience has learned that this company's axes require checking, every subsequent chart is read with suspicion, including the honest ones, and the analyst who spent a year building careful views pays the cost of a colleague's ninety-minute deck. The practice lesson is that chart credibility is organizational rather than individual, and it is consumed by whoever spends it.

The third vignette is the dashboard graveyard, and it is the vignette most likely to describe a student's first job. It is likewise a composite based on recurring organizational practice rather than a citation of published usage statistics. Organizations that deployed self-service BI at scale generally built far more dashboards than anyone uses, and the ones that later instrumented their platforms — most BI tools log which workbooks are opened, by whom, how often, and for how long — commonly report distributions that are uncomfortably skewed: a small minority of dashboards carrying nearly all of the viewing, a long tail opened once by their author and never again, and a substantial population of workbooks whose only regular visitor is the automated refresh job. The diagnosis, in the retrospectives that followed, was rarely that the charts were badly drawn. It was that they had been built without a decision attached — commissioned as "visibility into X," delivered as twelve views of X, and abandoned because nobody's Tuesday actually required any of them. The teams that responded well did two things that connect directly to this guide's spine. They began requiring a named decision and a named decision-maker before a dashboard was built, which is the analytic specification of Section 2.5 applied to a visual deliverable. And they began treating usage telemetry as a feedback loop, retiring unused workbooks on a schedule and interviewing the users of the surviving ones, which turned dashboard-building from a request-fulfillment queue into something closer to product work. The practice lesson is that the failure rate of visualization work in industry is dominated not by encoding errors but by unattached artifacts — beautiful charts nobody needed — and that the defense is upstream of anything in this chapter.

In your first analyst job, these three compress into one expectation, and it is this chapter's closing thread: analysts are judged as much on the charts they refuse to make as on the ones they build. Some of the refusals are the obvious ones — the pie with eleven slices, the truncated bar, the dual axis requested specifically because the two lines "look better together." Most are quieter. The refusal to show a metric without the reference it must be read against. The refusal to build the fourteenth view of a dashboard nobody opens. The refusal to put a magnitude comparison on a map because the map is what the executive asked for. The refusal to let a five-minute exploratory chart, correct for its purpose, be pasted into a deck where its defaults become claims. None of these produce a deliverable, and all of them are visible to the people who matter, because a colleague who says "I can build that, and here is what it would make people believe that is not true, so here is what I would build instead" is doing something a chart-generating tool cannot do and an assistant will not do unasked. That sentence is the job. The tool draws; the assistant drafts; the analyst decides what should be seen, and takes responsibility for what the seeing produces.

12.16 Ethics, Misleading Mechanics as an Integrity Obligation

Section 12.15 ended on the charts an analyst declines to build. This section examines why declining is an obligation rather than a preference, and it extends the guide's running ethics discussion — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), causal language (Section 7.15), acting on predictions about people (Section 8.15), fairness across groups (Section 9.15), forecast accountability (Section 10.15), and experimentation on customers (Section 11.15) — into the one medium where a false impression can be produced without a single false statement.

That is the feature which makes visualization ethics distinctive, and it should be stated at full strength. Every prior chapter's ethical failures involved something an analyst wrote, chose, or withheld that could in principle be quoted back: a metric defined to flatter, an exclusion left undocumented, a causal verb attached to correlational evidence, a forecast stripped of its interval. The mechanics of Section 12.7 involve none of that. The deck circulating since Thursday contains no false number, no misdefined metric, no overclaiming sentence, and no undocumented exclusion. It contains an axis that starts at $240,000. Every element of it would survive a fact-check, and its lead chart nonetheless caused a director of real estate to believe something about StyleCraft's suburban

business that is not true. An ethics built entirely on the accuracy of statements does not reach that case, which is why the profession needs one built on the accuracy of impressions.

CONCEPT

The Default Is Not a Defense

Four of the five mechanics in Table 12.3 are available as software defaults or single clicks, and this fact is usually offered in mitigation. It is the opposite. A defense of the form "the tool chose the axis" concedes precisely the thing that matters, which is that nobody looked — and once an analyst has read this chapter, "nobody looked" is a description of the analyst rather than of the tool. The obligation this creates is small and absolute: before any view leaves your screen, read its axis floor, name its grain, and state the comparison a reader will make. That is under a minute of work, it catches the great majority of what Section 12.7 catalogs, and it converts the standard from *did I intend to mislead* — a question about the analyst's interior state, which nobody else can check and which is therefore useless as a professional standard — to *did I check*, which leaves evidence and can be audited. The distinction matters most in the case that actually occurs. Almost nobody in marketing sets out to build a deceptive chart. A great many people ship one, quickly, under deadline, because the default filled the frame and the deadline filled the day, and the injury to the reader is identical either way.

Three obligations follow, and they run from the narrow to the broad.

The first is the duty to inspect one's own output, which is what the Appendix E checklist exists to make routine. An obligation that depends on remembering to care will be honored on quiet weeks and abandoned on the weeks that matter, which are exactly the weeks when charts reach decision-makers. Making it a checklist — read the axis, name the grain, name the comparison, run the grayscale check, write the footer — removes it from the domain of virtue and puts it in the domain of process, which is where obligations survive.

The second is the duty to correct other people's charts, and it is harder because it is social. The merchandising analyst who built the circulating deck is a colleague, junior, and under the same deadline; the director of real estate has already praised it; and the analyst who raises the axis question is choosing a small, immediate, personal cost in order to prevent a larger, deferred, institutional one. This is the ordinary shape of professional integrity and it is worth naming as such, because students consistently underestimate how often analytical ethics presents itself as an awkward conversation rather than as a dilemma. Two things make the conversation survivable, and both were built into Lab 12.2 Part B. Name the mechanic rather than the person, since "this axis starts at 240,000, which makes a 4.6 percent change read as a twelvefold jump" is a fact about a chart that anyone can verify, while "your chart is misleading" is a claim about a colleague that invites defense. And bring the repair, because a correction that arrives with a rebuilt chart attached is a contribution and one that arrives alone is a complaint.

The third is the duty that attaches to the audience rather than to the chart, and it is the one that will grow over these students' careers. Readers of a marketing chart are almost never in a position to check it. They do not have the extract, they do not know the grain, they cannot tell whether the window was chosen or inherited, and they are reading in a meeting with eleven other items on the agenda. Every one of those asymmetries transfers responsibility to the person who built the view, and the transfer is total in the case that Chapter 13 will inherit: a chart that travels beyond the room where its author can be questioned. The working standard this guide adopts is therefore a disclosure standard rather than an intention standard. Build the view so that the choices which could have gone another way are visible on it — the axis range, the window, the grain, the filters, the reference — and an audience that cannot check your work can at least see what you decided. That is what the footer discipline of Section 12.9 is actually for, and it is why five lines of small type at the bottom of a chart are an ethical instrument rather than a formatting convention.

One boundary closes the section, because students reasonably ask where the line sits and this chapter is not the place where it is drawn. Emphasis is legitimate. Choosing to color the suburban series and gray the others, choosing

to sort by the value under discussion, choosing to annotate the opening date — all of these direct a reader's attention, all of them are decisions the analyst makes on the reader's behalf, and none of them misstates a quantity. Distortion is different in kind: it changes what the reader believes the numbers are. This chapter owns the mechanics by which the second happens and asserts the obligation to prevent them. Where exactly emphasis becomes distortion, in a chart built to persuade an executive who has ten minutes and a decision to make, is the harder question, and it belongs to Chapter 13, which will have to answer it in front of an audience rather than in a lab.

12.17 Chapter Summary

This chapter opened Part III by treating visualization as an instrument rather than as a finish, and by claiming the first of its two jobs. A chart in its analytical role is built for discovery — dense, disposable, drawn so that the analyst can find out what the data contains — and a chart in its communicative role is built for transfer, and the properties that make each good actively damage the other, which is why the pre-read due Monday and the dashboard due in December are different objects rather than different drafts. Underneath both sits a grammar: every chart is a set of marks, a set of assignments from data fields to visual channels, and the axes, aggregations, and sort orders those channels are read against, so that a chart can be specified in a sentence, audited component by component, and altered one component at a time. The grammar becomes consequential through the perceptual hierarchy, which is the chapter's central empirically measured ordering — position read most accurately, then length, then angle, then area, then color intensity — and which converts chart selection from taste into a prediction about how accurately a reader will decode the comparison she is being asked to make. From there the chapter made the choices operational. Charts were selected by analytic purpose rather than from a gallery, with distribution, comparison, trend, relationship, composition, and geographic mapped to the StyleCraft questions Chapters 4 through 10 had already asked. Scales were examined as claims about what visual distance means, with the logarithmic axis recovered as the correct instrument for comparing growth rates across very different levels and the zero-baseline obligation reassigned from blanket rule to encoding-specific duty: an ordinary bar whose full length encodes magnitude must start at zero, a floating or interval mark must start at the reference its length is measured from and say so, an area mark must map zero to zero area, and any mark whose position carries the value needs a range that was chosen and stated rather than inherited from the software. The five standing mechanics were then cataloged and repaired — axis truncation, which on StyleCraft's own quarterly numbers drew the second bar twelve times as tall as the first for a value only 1.046 times as large, and which experimental work shows survives both correct labeling and explicit warning; dual axes, whose apparent coupling is manufactured by two arbitrary range choices and which invites a causal reading that Section 7.5 already disposed of; the cherry-picked window, defeated by showing a full seasonal cycle and stating why the window starts where it starts; area distortion, defeated by scaling area rather than radius and by moving magnitude comparisons off the map entirely; and the three-dimensional effect, which has no analytical use. Color completed the encoding system as a system: palette type chosen from the field's level of measurement per Section 3.4, color spent on emphasis rather than spread across every category, accessibility treated as a condition of correctness, screened by a grayscale conversion and then confirmed with a color-vision-deficiency simulation and a check that no essential meaning rests on color alone. Reference lines and annotation then supplied what no encoding can, which is something to read the values against — the visual form of the baseline discipline of Section 2.7 — along with the footer that records source, window, filters, and grain and thereby keeps a chart interpretable after it separates from its author. The tool arrived last and mapped onto what students already owned: Tableau's dimensions and measures closely parallel the variable roles of Section 3.5 while remaining software roles that govern query construction, discrete and continuous determine whether a field produces headers or an axis, and aggregation in the view is where the grain discipline of Section 3.3 becomes invisible — which the labs demonstrated by showing StyleCraft's entire suburban finding disappear into a 3 percent difference when average order value is computed from line-grain rows and reappear as a 66 percent difference when it is computed at order grain, with no error message in either case. The labs built the expansion story six ways from two certified extracts that reconcile to the cent, and then produced the chapter's most useful single exhibit: seventeen small multiples on one shared scale, plotted first against calendar time, where the newest stores look like failures, and then against store age, where they are read against the band their predecessors traced — the same data, the same facets, one changed

axis, and a recommendation that may not survive the change. The AI section installed the chapter's verification theme as a routine that predicts an AI-drafted chart's faults from its specification before the chart is rendered, on the grounds that a picture is persuasive in a way a specification is not, and named the nine failure modes an assistant reliably produces, chief among them the axis truncated on request and the per-order metric computed from per-line rows. And the ethics section drew the distinction that makes this chapter's obligations unusual: every mechanic in it produces a false impression without a false statement, so the professional standard cannot be *did I intend to mislead* but must be *did I check*, which is a minute of work and leaves evidence.

Looking ahead, the pre-read now exists, and its readers will spend the next week disagreeing with it, which is what it was built for. On December 2 the disagreement ends and a decision begins, and the artifact that meeting runs on cannot be this one. The next chapter turns to data visualization for communication and decision-making, where the reader is a chief executive with ten minutes rather than a colleague with a week, where the three-second test replaces the visual audit as the governing constraint, and where every technique this chapter deliberately withheld — assertion titling, aggressive decluttering, narrative sequencing, interactivity that guides rather than explores — becomes the subject. The mechanics of Section 12.7 travel forward unchanged and are not re-argued there; what Chapter 13 must add is the harder line this chapter declined to draw, between the emphasis that helps a decision-maker see what matters and the concealment that helps her agree. Between here and there sits Project #2, whose brief the next chapter releases and whose deliverable is the dashboard Chapter 13 teaches. The exercises below prepare the certified extracts and exploratory views that students may reuse once that brief arrives.

12.18 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: name the analytic purpose before choosing a form, state what one mark represents before building a view, read the axis floor on anything encoded by length, spend color on emphasis rather than on decoration, put the reference the reader needs on the chart itself, and predict what a chart will show before rendering it. Not everything below is required. Core practice, which every student should complete: 12.18.1, 12.18.2, 12.18.3, and 12.18.4. Homework options, from which your instructor will assign a subset: 12.18.5 and 12.18.6. Instructor-led discussion, prepared for class rather than submitted: 12.18.7 and 12.18.8. Each exercise also carries its label individually. No new project deliverable is due this week; the two Homework Submission items are assignment options from which your instructor will select, and any of them that is assigned is graded. Project #2's brief is released in Chapter 13, and Section 12.18.5 produces the certified extracts and the first views that project will build on, so students who complete it will start Project #2 a week ahead.

12.18.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- Distinguish visualization's analytical job from its communicative job, and name one property that improves a chart for the first job while damaging it for the second.
- Define a mark and an encoding, and specify a sorted bar chart of revenue by store in that vocabulary, naming every component including the ones the software supplies by default.
- State the perceptual hierarchy in order, and explain why a pie chart of seven categories is a worse instrument than a sorted bar chart of the same seven.
- Explain why ordinary magnitude bars must begin at zero while position-encoded lines need not, in terms of what each mark encodes rather than in terms of convention, and state what a floating or interval bar must begin at instead.
- A colleague defends a truncated bar chart on the grounds that the axis is clearly labeled. State the empirical finding that answers him, and cite the study.
- Explain what makes a dual-axis chart's apparent coupling arbitrary, and name a replacement that answers the same question honestly.
- State the rule for matching a palette type to a field, and give one StyleCraft example of each of the three palette types.

- Explain what a reference line does that no encoding can do, and connect it to the discipline introduced in Section 2.7.
- Define aggregation in the view, and explain how it can produce a correct number that answers the wrong question with no error message.
- Explain what small multiples buy and what they cost, in terms of the perceptual hierarchy, and give the StyleCraft case in which the purchase was clearly worth the price.

12.18.2 Predict the Encoding (Required Practice)

For each of the following comparisons, state (a) the analytic purpose, (b) the encoding channel you would use to carry the comparison, (c) that channel's rank in the perceptual hierarchy, and (d) one encoding you considered and rejected, with the reason. Do not build anything.

- Which of StyleCraft's seventeen stores produced the most trailing-twelve-month revenue, and by how much more than the second.
- Whether the share of revenue coming from occasionwear differs between NYC Urban and NYC Suburban customers.
- Whether weekly revenue at the Washington store is growing faster than weekly revenue at SoHo, given that SoHo runs at roughly $7,900 a week and Washington at roughly $1,800.
- Whether customers with the app spend more than customers without it, across the full range of spend rather than on average.
- Whether the eight Wave 4 stores are clustered in any particular part of the country.

Then answer the question the exercise is actually about. For the third item, state which scale you chose and write the one-sentence caption that would prevent a reader from misreading the level difference. For the fifth, state what your chosen view can and cannot establish about why those stores were opened where they were.

12.18.3 Purpose First (Required Practice)

Return to Table 12.6 and complete it for the three views the labs did not build — A, D, and E — filling every column: purpose, the comparison the reader must make, the grain of one mark, and the reference the view must carry. Then, for each of the three, write a full specification in the form Section 12.12's request prompt asks an assistant to produce: mark type, every field-to-channel encoding, the aggregation and grain, the axis ranges with a justification for each, the sort order, and the palette with its type. Finally, for View E, name two chart forms that could answer the composition question, state which you would build for the pre-read and which you would build for yourself while exploring, and explain the difference in terms of Section 12.2.

12.18.4 The Grain Trap (Required Practice)

Using `transactions_enriched` in Tableau or in Colab, whichever you prefer:

Compute average order value for NYC Urban and NYC Suburban customers two ways — as `AVG([line_revenue])` on the line-grain file, and as `SUM([line_revenue]) / COUNTD([order_id])` — and report all four numbers. State which pair is average order value and which pair is something else, and name the something else precisely. Then compute the difference between the metros under each definition, in dollars and as a percentage, and write the one sentence a reader would take away from each, so that the two sentences sit side by side on the page.

Next, decompose the correct gap. Compute items per order and average line value for each metro, and show that the gap equals the basket-size component plus the price-and-mix component, with the arithmetic displayed. State in one sentence what the decomposition establishes about whether StyleCraft charges suburban customers more, and connect the finding to what the course data dictionary documents about the dataset's design.

Finally, list three other metrics used anywhere in Chapters 5 through 11 that require a grain other than order-line, name the grain each requires, and state where in a Tableau workflow you would enforce it — in Prep, in a calculated field, or in the view's dimensions — with one sentence of justification.

12.18.5 The Visual Audit Deliverable (Homework Submission)

Build the pre-read described in Section 12.14, as a Tableau workbook plus a two-page written guide.

The workbook must contain, in order: the two certified extracts from Lab 12.1 Part A with the revenue reconciliation documented; the six views of Table 12.6, each carrying its reference and its footer; and the two faceted views of Lab 12.1 Part C, on calendar time and on store age. Every view must state, in its caption, what one mark represents.

The written guide must contain the reading guide paragraph, one paragraph per view naming the question it answers and the finding it supports, the two-picture argument about store age with an explicit statement of what changed between the pictures, a paragraph on what the map does not establish, and an appendix listing at least three charts you built and rejected with one sentence apiece on why. Include, as the guide's final paragraph, the single sentence you would say first if you had thirty seconds with the director of real estate — and note that Chapter 13 will ask you to rebuild the whole thing around that sentence.

12.18.6 AI Chart Audit (Homework Submission)

Run the five-step routine of Section 12.12's AI in Practice box on a question of your choosing from Table 12.6, and document it per Appendix D. Submit: (a) your request prompt and the assistant's three candidate specifications, with the schema you pasted and any column the assistant invented despite it; (b) your written predictions, made from the specifications before building anything, naming for each candidate the comparison, the channel, its hierarchy rank, and the two faults you expected; (c) the built views, faults included, with screenshots; (d) the completed audit against the three transferable points of Table 8.5 and all three of Table 12.5, with each fault named, quoted, and quantified where quantification is possible; (e) your grade of your own predictions, including both the faults you missed and the ones you predicted that did not occur, with a paragraph on what the pattern reveals about your model of the assistant; and (f) the repaired views with a one-sentence note per repair recording what changed in the reading.

Your grade rests on the audit and the prediction, not on the assistant's output. An exchange in which the assistant produced four faults and you predicted three of them is a better submission than one in which it produced a clean chart you did not have to inspect.

12.18.7 Spot the Mechanic (In-Class Discussion)

Each of the following describes a chart you will meet in commercial practice. Name the mechanic from Table 12.3 or the audit point from Table 12.5 that it violates, state in one sentence what a reader will believe that is not true, and state the single change that would repair it.

- A quarterly revenue bar chart whose vertical axis runs from $4.2 million to $4.6 million, with both values labeled on the bars.
- A line chart showing email sends and revenue on twin vertical axes, presented with the caption "email drives revenue."
- A map of the United States shaded by total orders, presented as evidence that the Northeast is StyleCraft's strongest market.
- A bar chart of average order value by loyalty tier, sorted alphabetically as None, Insider, VIP, Backstage, in four saturated hues.
- A chart of campaign performance covering the six weeks following a campaign launch, in a business whose fourth quarter carries a documented holiday lift.
- A dashboard tile reading "Average order value: $50.41," computed on a file with one row per order line.

For the last item, state what the correct number is and estimate what the error would do to a revenue projection built on it.

12.18.8 The Colleague's Deck (In-Class Discussion)

You are the analyst in Section 12.1. The merchandising analyst's deck has reached the director of real estate, who has replied "this is great, let's use these." You have four working days before the pre-read circulates, and the

December 2 meeting is three weeks out. The merchandising analyst reports to a different manager, is two years into her first job, and built the deck in ninety minutes using an AI assistant because she was asked to produce something by Friday.

Prepare a position, and address each of the following explicitly. What do you send, to whom, and in what order — the merchandising analyst first, her manager first, the director first, or all three at once? What exactly does the message contain, given Section 12.16's instruction to name the mechanic rather than the person and to bring the repair? How do you handle the fact that the director has already endorsed the charts publicly? Which of the four faults would you raise and which, if any, would you let stand, and on what principle do you distinguish them? What would you do differently if the deck had already gone to the chief executive rather than to a peer of yours? And finally: the merchandising analyst did nothing that this course had taught her not to do, and produced in ninety minutes something that took you two labs to check. State what organizational change — not what individual virtue — would have prevented the situation, and name the section of this chapter that describes it.

12.19 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Aggregation in the view. Tableau's behavior of computing each measure at the level of detail defined by the dimensions currently in the view, recomputed silently on every change, so that a metric requiring a specific grain will be computed at the wrong grain without warning whenever the view's dimensions do not match it (adapted from Murray, 2016, and the grain discipline of Section 3.3).

Analytic purpose. The class of question a view is built to answer, stated as the comparison or structure the reader must extract rather than as the chart form that displays it; the six purposes covering nearly all marketing analysis are distribution, comparison, trend, relationship, composition, and geographic (adapted from Shneiderman, 1996; Munzner, 2014).

Analytical visualization. A chart whose dominant job is discovery — built by the analyst, for the analyst, to find out what the data contains, and judged by whether it reveals structure not already known; typically dense, disposable, and labeled enough to be reproduced rather than polished for an outside reader (Section 12.2; Tukey, 1977; Gelman & Unwin, 2013).

Annotation. Explanatory text placed on a chart itself — a label on a mark, a note on a region, or a footer recording source, window, filters, and grain — supplying context the encoding cannot carry and keeping the chart interpretable after it separates from its author (adapted from Tufte, 2001; Few, 2012).

Area distortion. Misencoding of a quantity in the size of a two-dimensional mark, most often by scaling radius or side length in proportion to value so that area grows with the square of the value; compounded by readers' general underestimation of area differences, and repaired by scaling area to value, labeling a size legend, and moving precise comparisons to a length- or position-encoded view (adapted from Cleveland & McGill, 1984; Tufte, 2001).

Axis truncation. Omission of the zero origin from an axis whose marks encode value by length or area, so that visual proportion no longer corresponds to numerical proportion; the distortion survives correct axis labels and explicit warnings, and is legitimate only for position-encoded marks, where it restricts the view rather than misstating the proportion (adapted from Pandey et al., 2015; Correll et al., 2020).

Cherry-picked window. Selection of the displayed extent of data — of time, of categories, or of units — so that the visible pattern differs from the pattern in the full data; every displayed number may be correct. The signature is a window whose start has no stated justification, and the repair is to state why the window begins where it begins and to show at least one full seasonal cycle in a seasonal business (Section 12.7; Section 10.3).

Communication visualization. A chart whose dominant job is transfer — built by the analyst, for someone else, to convey a finding already established, and judged by whether a reader who has never seen the data arrives at the intended understanding in the time available; typically sparse, durable, and finished, and the subject of Chapter 13 rather than of this chapter (Section 12.2; Gelman & Unwin, 2013).

Dimension. In Tableau's data model, a field used to partition, group, or slice the data, defining the categories, dates, and identifiers at which values are separated and thereby setting the level of detail of a view; closely parallel

to the categorical and identifying variable roles of Section 3.5, but a software role that can be changed without changing what the field means (adapted from Murray, 2016; Tableau, 2026a).

Discrete and continuous fields. A discrete field takes separate values and produces headers or panels; a continuous field takes values along an unbroken range and produces an axis. The distinction is independent of whether a field is a dimension or a measure, and determines whether marks are separated into panels or positioned along a scale (adapted from Murray, 2016).

Dual axis. A chart plotting two measures against a shared horizontal axis using two independently scaled vertical axes, so that the series' apparent alignment or divergence is a property of the arbitrary scaling rather than of the data; defensible only when the two scales are fixed by definition, and otherwise replaced by stacked panels or by indexing both series to a common base (adapted from Cairo, 2019; Few, 2012).

Encoding. The assignment of a data field to a visual channel of a chart's marks — position, length, angle, area or size, color hue, color intensity, shape, or texture — which together with the mark type and the scales fully specifies a chart (adapted from Bertin, 1983; Wilkinson, 2005).

Faceting. Partitioning data by a dimension and drawing one panel per value of that dimension, producing small multiples; the operation that trades common-baseline comparison for the ability to compare many subsets at once (adapted from Tufte, 2001; Munzner, 2014).

Linear scale. An axis on which equal visual distances represent equal differences, so that the question the axis answers is "how much more"; the default for nearly every chart form and the required scale wherever a mark's length or area encodes magnitude (Section 12.6).

Logarithmic scale. An axis on which equal visual distances represent equal ratios, so that the question the axis answers is "how many times more" and, for a series against linear time, at what proportional rate a quantity is growing; a straight line indicates approximately constant exponential growth. Available only when every plotted value is positive, since zero and negative values cannot be placed on a log axis; inappropriate for length-encoded marks, and unfamiliar enough to most executive readers that its use requires an explicit note (Section 12.6; Tableau, 2026e).

Mark. The geometric object a chart draws to represent data — point, line, bar, rectangle, circle, arc, or filled area — whose type determines what kind of object the reader is looking at and which visual channels are available to encode fields (adapted from Bertin, 1983; Wilkinson, 2005).

Measure. In Tableau's data model, a quantitative field aggregated by default within the partitions defined by the dimensions present in a view; closely parallel to the metric and quantitative variable roles of Section 3.5, but a software role, and one a field can be converted out of (adapted from Murray, 2016; Tableau, 2026a).

Perceptual hierarchy. The empirically measured average ordering of visual encoding channels by the accuracy with which readers extract quantitative values from them — position on a common scale, then position on identical non-aligned scales, then length, angle and slope, area, volume, and color intensity — used to assign the most important comparison to the highest-ranking available channel. A robust average ordering for elementary quantitative judgments rather than a universal ranking: measured performance varies with task, display, reader, and context, and the ordering governs the comparison the layout actually asks the reader to make (adapted from Cleveland & McGill, 1984; Heer & Bostock, 2010).

Reference line. A line, band, or marker drawn at a separately defined benchmark — external or data-derived — against which the plotted marks are interpreted, supplying the comparison the reader would otherwise have to provide from memory. Externally supplied decision thresholds should be declared before results are inspected; descriptive references such as an average or a median may be computed from the plotted data but must be labeled as such. The visual form of the baseline discipline of Section 2.7 (adapted from Few, 2012; Munzner, 2014; Tableau, 2026d).

Sequential, diverging, and categorical palettes. Sequential palettes vary lightness within one hue and encode ordered quantities; diverging palettes run two hues outward from a meaningful midpoint and encode distance and direction from a reference; categorical palettes assign distinguishable hues of roughly equal lightness to unordered groups and carry no magnitude information. Matching palette type to the field's level of measurement is a correctness requirement rather than a stylistic one (adapted from Harrower & Brewer, 2003; Munzner, 2014).

Small multiples. A series of small charts sharing identical axes, scales, and encodings, each showing the same measures for a different subset, arranged so the subsets can be compared by scanning; because the scale is shared, differences between panels are read as differences in the data (adapted from Tufte, 2001; Munzner, 2014).

12.20 Further Readings

Students who want additional background may begin with the following readings. The design treatments are listed first, the perceptual evidence second, and the tool reference last.

- Cairo (2019) for the most readable modern treatment of how charts mislead, written for a general audience by a journalist and academic who has spent a career on the problem; its chapters on scale and on charts that appear to show causation are the direct expansion of Sections 12.6 and 12.7.
- Few (2012) for the practitioner's standard on business charts and tables — the source of much of the reference-line, annotation, and emphasis discipline used in Sections 12.8 and 12.9, and the most immediately applicable book on this list for a marketing analyst.
- Munzner (2014) for the systematic treatment of visualization as a design problem, organized around what task a visualization must support and which channels can support it; the level above this chapter's Sections 12.3 through 12.5, and the best available bridge from these ideas to the more advanced visualization work that follows this guide.
- Cleveland and McGill (1984) for the original experiments behind the perceptual hierarchy — short, readable, and worth reading in the original, because seeing how the ordering was actually measured is what makes it a fact rather than an opinion to be argued with.
- Correll et al. (2020) for the modern experimental treatment of axis truncation, including the uncomfortable finding that warning readers does not repair the distortion; the paper behind Section 12.7's insistence that the repair must be structural.
- Murray (2016) together with Tableau Help (2026a through 2026h) for the conceptual and the current-interface treatments of the tool. Read Murray for the account of the data model, dimensions and measures, and the discrete-continuous distinction that Section 12.10 maps onto Chapter 3's variable roles; read the documentation for the authoritative current build paths behind every instruction in Section 12.13 — field roles, Prep aggregation and joins, Prep functions, reference lines, axis configuration, geographic roles, proportional-symbol maps, and quantitative mark sizing. Where a printed source and the vendor's documentation disagree about an interface, the documentation governs, and it is the reference an analyst should learn to check first.

12.21 References

Amar, R., Eagan, J., & Stasko, J. (2005). Low-level components of analytic activity in information visualization. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (pp. 111–117). IEEE. https://doi.org/10.1109/INFVIS.2005.1532136

Bertin, J. (1983). *Semiology of graphics: Diagrams, networks, maps* (W. J. Berg, Trans.). University of Wisconsin Press. (Original work published 1967)

Borland, D., & Taylor, R. M. (2007). Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications, 27*(2), 14–17. https://doi.org/10.1109/MCG.2007.323435

Cairo, A. (2019). *How charts lie: Getting smarter about visual information*. W. W. Norton.

Cleveland, W. S., & McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association, 79*(387), 531–554. https://doi.org/10.1080/01621459.1984.10478080

Correll, M., Bertini, E., & Franconeri, S. (2020). Truncating the y-axis: Threat or menace? In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376222

Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten* (2nd ed.). Analytics Press.

Gelman, A., & Unwin, A. (2013). Infovis and statistical graphics: Different goals, different looks. *Journal of Computational and Graphical Statistics, 22*(1), 2–28. https://doi.org/10.1080/10618600.2012.761137

Harrower, M., & Brewer, C. A. (2003). ColorBrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal, 40*(1), 27–37. https://doi.org/10.1179/000870403235002042

Heer, J., & Bostock, M. (2010). Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 203–212). Association for Computing Machinery. https://doi.org/10.1145/1753326.1753357

Munzner, T. (2014). *Visualization analysis and design*. CRC Press.

Murray, D. G. (2016). *Tableau your data! Fast and easy visual analysis with Tableau Software* (2nd ed.). Wiley.

Pandey, A. V., Rall, K., Satterthwaite, M. L., Nov, O., & Bertini, E. (2015). How deceptive are deceptive visualizations? An empirical analysis of common distortion techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (pp. 1469–1478). Association for Computing Machinery. https://doi.org/10.1145/2702123.2702608

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages* (pp. 336–343). IEEE. https://doi.org/10.1109/VL.1996.545307

Tableau. (2026a). *Dimensions and measures, blue and green*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/datafields_typesandroles.htm

Tableau. (2026b). *Aggregate, join, or union data*. Tableau Prep Help. Retrieved July 30, 2026, from https://help.tableau.com/current/prep/en-us/prep_combine.htm

Tableau. (2026c). *Date properties for a data source*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/date_properties.htm

Tableau. (2026d). *Reference lines, bands, distributions, and boxes*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/reference_lines.htm

Tableau. (2026e). *Edit axes*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/formatting_editaxes.htm

Tableau. (2026f). *Create maps that show quantitative values in Tableau*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/maps_howto_symbol.htm

Tableau. (2026g). *Format geographic fields in Tableau*. Tableau Help. Retrieved July 30, 2026, from https://help.tableau.com/current/pro/desktop/en-us/maps_geographicroles.htm

Tableau. (2026h). *Tableau Prep functions reference*. Tableau Prep Help. Retrieved July 30, 2026, from https://help.tableau.com/current/prep/en-us/prep_function_reference.htm

Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Graphics Press.

Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley.

Wainer, H. (1984). How to display data badly. *The American Statistician, 38*(2), 137–147. https://doi.org/10.1080/00031305.1984.10483186

Wickham, H. (2010). A layered grammar of graphics. *Journal of Computational and Graphical Statistics, 19*(1), 3–28. https://doi.org/10.1198/jcgs.2009.07098

Wilkinson, L. (2005). *The grammar of graphics* (2nd ed.). Springer.

Wong, B. (2011). Points of view: Color blindness. *Nature Methods, 8*(6), 441. https://doi.org/10.1038/nmeth.1618

Zacks, J., & Tversky, B. (1999). Bars and lines: A study of graphic communication. *Memory & Cognition, 27*(6), 1073–1079. https://doi.org/10.3758/BF03201236

Chapter 13

Data Visualization for Communication and Decision-Making (Tableau II)

Audience, Hierarchy, Emphasis

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this chapter is licensed under CC BY 4.0.

Chapter Information

ABSTRACT

This chapter builds the communicative half of data visualization, using StyleCraft's go/no-go expansion decision as its running material and Tableau as its instrument. It establishes the reader as the binding design constraint through audience analysis, a stated three-second reading test, and the choice among exploratory and explanatory forms. The KPI hierarchy then spatializes the metric discipline of Section 3.5 into primary, supporting, and diagnostic tiers, and layout converts that hierarchy into reading order. Narrative arc and the assertion title turn views that display quantities into views that state findings, at the cost of an obligation to verify every sentence against the marks beneath it; emphasis and decluttering make those sentences findable; interactivity is bounded to navigation, and on this artifact that bound turns out to exclude filter controls entirely. Uncertainty receives its own treatment, expressed in decision units and drawn as regions whose outlines are computed upstream. Two labs build the executive dashboard, test it on a reader, and pair it with a recommendation memo.

KEYWORDS

data visualization; audience analysis; dashboard design; KPI hierarchy; assertion titles; preattentive attributes; decluttering; uncertainty communication; usability testing; Tableau

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Data visualization for communication and decision-making (Tableau II). In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Chapter 13, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Copyright © 2026 Jose Mendoza. Except where otherwise noted, this work is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0). You may share and adapt this material for any purpose, provided appropriate credit is given. Third-party trademarks, screenshots, figures, and other materials remain subject to their respective rights and licenses.

Tableau, Tableau Desktop, and Tableau Prep are trademarks of Salesforce, Inc. Google Colab is a product of Google LLC. ChatGPT is a product of OpenAI. Claude is a product of Anthropic. Gemini is a product of Google LLC. GitHub Copilot is a product of GitHub, Inc. NotebookLM is a product of Google LLC. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

COMPANION REPOSITORY

Datasets, notebooks, certified metric files, and figure sources for this chapter: https://github.com/jrmst102/businessanalytics

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this chapter, including literature discovery, source organization, outlining, preliminary drafts, prose revision, support for code and

analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the chapter's scope, learning objectives, methods, interpretations, and recommendations, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and code and analytical outputs were tested or otherwise reviewed for accuracy. Responsibility for the accuracy, originality, and final form of this chapter rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

Chapter Learning Objectives

By the end of this chapter, students should be able to:

85. Conduct an audience analysis for a specific visual deliverable, stating who will read it, what decision they must make, how long they will look, what they already know, and what would change their mind — and derive design consequences from each.
86. Apply the course's three-second reading test as a design constraint, predicting what a fresh reader will take away from an artifact before testing it, and revising the artifact rather than the reader when the prediction fails.
87. Distinguish exploratory from explanatory artifacts, choose among a dashboard, a report, and a data story for a stated decision and audience, and state the conditions under which a dashboard is the wrong deliverable.
88. Design an executive dashboard with an explicit KPI hierarchy — primary, supporting, and diagnostic — that spatializes the metric and KPI discipline of Section 3.5, so that a view's position and size encode its importance to the decision.
89. Lay out a dashboard for reading order, grouping, density, and whitespace, and justify each placement in terms of the sequence in which the reader must acquire the argument.
90. Construct a narrative arc across a set of views, write assertion titles that state what each view shows rather than what it plots, and verify every assertion against the underlying view before the artifact leaves the analyst's hands.
91. Use preattentive attributes deliberately to emphasize and de-emphasize, place annotation per Section 12.9 at the specific marks an assertion title depends on, apply data-ink discipline to remove what does not carry meaning, and explain why decluttering is a precondition for emphasis rather than an aesthetic preference.
92. Build interactivity that serves navigation and communication — filters with a stated scope, parameters used as navigation with titles that track them, tooltips, and dashboard actions — and identify the cases in which interactivity conceals the point rather than clarifying it.
93. Communicate the uncertainty established in Chapters 10 and 11 to an executive audience, constructing a prediction-interval ribbon from precomputed bounds at the forecast's own grain, and pairing the display with the asymmetric-cost reasoning of Section 2.7.
94. Pair a dashboard with a written recommendation that states the decision, the evidence, the uncertainty, and the conditions under which the recommendation would change.
95. Run a formative usability check on a visual artifact and a structured critique exchange with a peer, treating the reader's stated takeaway as data about the artifact, and document the resulting revisions.
96. Audit AI-generated titles, layouts, and recommendation prose against the views and evidence that must support them, and document the exchange per Appendix D.
97. State where emphasis becomes distortion in a persuasive artifact, defend a specific design choice against that standard, and identify the disclosures that keep a persuasive chart honest.

Chapter 12 built the analytical half of visualization and closed with an artifact designed to be argued with. The pre-read circulated on Monday, and by Wednesday morning it had done its job: three of the four readers replied, two of them at length, and the argument the December 2 meeting was going to have has now largely been had in writing. That is a better outcome than most analytical work achieves, and it is not the outcome this chapter needs. The pre-read is eight views, two of them faceted across seventeen store panels, every one carrying a footer recording source, window, filters, and grain — a document built for readers who have a week, a stake, and a reason to interrogate it. On December 2 the artifact will be read by a chief executive who has ten minutes, no prior exposure to the file, a

real-estate committee waiting on her, and eight leases that price on Friday. Nothing about the pre-read survives contact with that reader, and the temptation the whole chapter is written against is to believe that it might, because the analysis inside it is correct and correct analysis feels as though it ought to travel. It does not travel. It has to be carried, and carrying it is a separate discipline with its own techniques, its own failure modes, and its own ethical boundary — the boundary between helping a decision-maker see what the evidence shows and helping her agree with what the analyst has concluded. Chapter 12 owned the mechanics by which charts mislead and declined to draw that boundary. This chapter draws it, in front of an audience, under a deadline, which is where it actually has to be drawn.

CONCEPT

What This Chapter Is Really About

Chapter 12 said that a chart is a machine for making a comparison easy. A communication artifact is a machine for making a *decision* easy, and the two sentences are not the same sentence, because a decision requires more than a legible comparison. It requires the reader to know which comparison matters most, to know what the comparison implies for the choice in front of her, to know how confident the analyst is, and to know what would have to be true for the recommendation to be wrong — and it requires her to acquire all four in the order that makes them usable, in the time she actually has. Every technique in this chapter is a way of supplying one of those four things without requiring the reader to do work she will not do. The KPI hierarchy tells her which comparison matters most, by putting it where the eye lands first.

The assertion title tells her what the comparison implies, by saying it in a sentence instead of leaving her to infer it from an axis label. Emphasis and decluttering make the sentence findable by removing everything that competes with it. The uncertainty view tells her how much to trust it. And the written recommendation tells her what to do and under what conditions to stop doing it. The uncomfortable part, and the reason this chapter needs an ethics section that Chapter 12 could defer, is that every one of those techniques also works when the analyst is wrong. A confident hierarchy, a fluent title, and a clean layout will carry a mistaken conclusion into a decision just as efficiently as a correct one — more efficiently, in fact, because a mistaken conclusion is usually the simpler story.

Chapter 12's verification asked whether the chart showed what the data contained. This chapter's verification asks a second question on top of that one: whether the reader took away what the chart actually supports. Only the reader can answer it, which is why this chapter ends with a test on a human being rather than a check on a number.

13.1 Marketing Decision Context: The Go/No-Go Meeting

The pre-read went out Monday at 4:40 p.m. By Wednesday morning three of the four readers had replied, and the replies are the reason this chapter exists.

The head of merchandising wrote first and wrote briefly: the composition view is the whole argument, she has been saying so for a year, and she wants it in front of the chief executive. The director of real estate wrote second and at length, and his note is the one that mattered. He had opened with the calendar-time facet — the one showing the eight Wave 4 stores as short low lines at the right edge — and had written most of a paragraph arguing that the newest stores were not performing before he reached the store-age facet on the same page and, in his words, "had to start the email over." He is now the expansion's most careful advocate, which is exactly what a good pre-read produces. The finance partner from Chapter 11 replied third, with a single question that the pre-read does not answer anywhere in its eight views: *at what point does one of these stores cover its own cost, and how many of the eight sites look like the ones that got there?*

That question is the December 2 meeting. StyleCraft's chief executive convenes the review at 9:00 a.m. and has, on the agenda circulated Friday, allotted it fifteen minutes, of which the analyst should assume ten. Eight sites in the suburban and resort formats are under letter of intent — the fifth wave's flagship sites are not yet at that stage and are a separate decision on a separate benchmark; the letters of intent carry an eight-week option on any site StyleCraft does not execute on Friday, which is the fact that makes a conditional recommendation operationally available rather than rhetorical; the leases price at the end of the week; and the decision is a single binary with a conditional version available — commit to all eight, commit to none, or commit to a subset defined by a stated rule. The chief executive has not read the pre-read and will not. She has read the one-page summary the VP of Marketing wrote from it, she will arrive with the finance partner's question in her head because the finance partner reports to her, and she will make the call in the room.

The VP's brief to the analyst on Tuesday afternoon was four sentences long and is worth quoting because it is the actual design specification. One screen. Ten minutes. She should be able to say what she thinks after three seconds and then spend the rest of the time testing it. And she will ask what would have to be true for you to be wrong, so have that on the screen too, not in your head.

The first draft, built Tuesday night, does not meet any of those conditions, and the way it fails is instructive because it is the failure this chapter's students will reproduce on their first attempt. It was assembled the fast way, which is to say it was assembled out of the pre-read: the eight views of Lab 12.1 dragged onto a dashboard canvas, arranged four across and two down, each keeping the title Tableau generated from its field names — "SUM(Line Revenue) by Order Week and Metro," "Repeat-Order Share by Store Type" — each keeping the five-line analytical footer, each keeping its full legend, and the whole grid sharing a filter panel down the right side with seven controls on it. Everything in it is true. Every axis starts where it should, every grain is stated, every reference line is present and labeled, and the file it runs on is the certified extract that reconciles to the cent. It would survive any audit in Chapter 12, and shown to a reader for three seconds it communicates one thing, which is that a competent person has done a lot of work. Shown to the VP for rather longer than three seconds, it produced the question that names the problem: *what am I looking at first?*

There is no answer, because nothing on the canvas encodes an answer. Eight views of equal size in a regular grid assert that the eight matter equally, which is a claim the analyst does not believe and would not defend if asked; the largest visual object on the screen is the filter panel, which asserts that the reader's most important act is filtering; and the titles describe what each chart plots rather than what it shows, which leaves the reader to derive eight findings in ten minutes from material she has never seen. The draft is not wrong. It is unstructured, which for a ten-minute reader is a more expensive failure than being wrong would be, because a wrong chart gets caught in the room and an unstructured one just quietly fails to land and everyone blames the meeting for running long.

There is also a second problem, and it is the harder one. The finance partner's question does not have a clean answer, and answering it at all requires two numbers that the pre-read never needed and that no earlier chapter produced. The finance partner supplies the first as a planning assumption: the annual fixed footprint of a new StyleCraft store in the suburban and resort formats — rent, staffing, and allocated overhead — is $92,000, which is $1,769 of contribution that a store must generate in an ordinary week to carry its own fixed cost. The certified extract supplies the second: the contribution margin on the suburban and resort store population, computed as `line_revenue` less `line_cost` over `line_revenue`, runs at 46 percent, so the weekly *revenue* at which such a store covers its footprint is approximately $3,846. Against that declared threshold, the four mature suburban and resort stores first reached weekly coverage on a four-week average in weeks 43, 46, 51, and 57 of their operating lives — later than the weeks in which each first crossed the line once, which were 38, 42, 44, and 51, and the gap between the two is itself part of the answer, since a store whose two weeks are far apart crossed on a single week its neighbors did not repeat. The eight Wave 4 stores, all of them suburban or resort, are seventeen weeks old. Two of them are tracking above the range their predecessors occupied at the same store age, five are inside it, and one, Scottsdale, has been below it since week six. Section 13.12 states where every one of those figures comes from, which population it describes, and requires the student to derive it before any title uses it, because a number that arrives in a chapter without a derivation is a number a student cannot check.

Chapter 10's forecast supplies the last of the material, and it has to be rebuilt before it can be used. Chapter 10 produced a six-month planning forecast of chain *revenue*, because that was what the merchandising plan consumed.

A lease decision spans a planning year and turns on contribution rather than revenue, so this chapter reruns Chapter 10's model-selection procedure on the daily contribution series — the same validation design, not the same winner assumed to transfer — and reports the fiscal year the leases will live in. Two properties of the result matter in the room. It is an **outlook rather than a forecast**: July through November have already happened by December 2 and are carried as actuals, so only the seven months from December onward have a range around them. And it is stated in **contribution**, which means the interval needs no margin applied to it before it can be compared with anything. StyleCraft's FY2027 chain contribution is therefore $0.51 million already banked plus a central estimate of $0.74 million still to come, or $1.25 million, with the part still open running from $0.62 million to $0.86 million. The width of that range is not a technicality: end to end it is worth roughly two mature stores' annual contribution. The honest answer to the chief executive's question is therefore conditional, and conditional answers are the ones that communicate worst — they are the ones that get compressed in the retelling, that get heard as *the analyst is not sure*, and that tempt the analyst into stating something cleaner than the evidence supports in order to be useful. Section 13.9 is about that temptation and Section 13.15 is about where giving in to it stops being a communication choice and becomes an ethical one.

So the commission is a single artifact with an unusual specification: one dashboard, readable in three seconds at the top level and interrogable for ten minutes below it, carrying a recommendation that is conditional without being evasive, accompanied by a written memo that says what the screen cannot. Building it is Lab 13.1. Testing whether it works — on an actual reader, before the meeting rather than during it — is Lab 13.2, and it is the chapter's verification theme, because the only evidence that a communication artifact communicates is what a person who reads it says afterward.

13.1.1 Opening Case Questions

Keep these questions in mind while reading, and return to them after completing the labs.

- The first draft contains every finding in the pre-read and is correct in every particular. Write one sentence stating what it fails to do, and one sentence explaining why "the information is all there" is not a defense for a ten-minute reader.
- The chief executive will look at the screen for roughly three seconds before forming a first impression. Name the one sentence you would want her to form, and then name the design decision — position, size, color, or title — that you would use to produce it.
- The finance partner's question has a conditional answer: some of the eight sites resemble stores that reached weekly coverage and some do not. List two ways to communicate a conditional recommendation without either overstating confidence or sounding as though you have no recommendation, and predict which section of this chapter each comes from.
- A dashboard shows eight gray lines of equal weight on a single set of axes. You want the reader to find one particular line without searching for it. Name the single visual property you would change, state why changing two properties at once would work less well, and predict which section of this chapter explains the answer.
- The pre-read was built to be disagreed with. The dashboard is built to support a decision. Name one thing the dashboard must add that the pre-read deliberately withheld, and one thing the pre-read carried that the dashboard should drop — and state, for the second one, what is lost by dropping it.

13.2 Know Your Reader

Section 13.1 described a dashboard that fails for a reason no audit of its charts would detect. This section supplies the diagnosis, and the diagnosis is that the draft was designed against the data rather than against a person. Chapter 12's standard for a view was internal — does the encoding support the comparison, is the axis honest, is the grain what the metric requires — and every one of those questions can be answered by the analyst alone at her own screen. This chapter's standard is external. It depends on a reader with a finite amount of attention, a specific decision, and prior knowledge the analyst does not control, and it cannot be answered by inspection. It can only be predicted and then tested, which is why the reader has to be specified before anything is built.

DEFINITION

Audience Analysis

Audience analysis is the explicit statement, made before an artifact is designed, of who will read it, what decision or action it must support, how much time and attention the reader will give it, what the reader already knows and believes, what technical vocabulary can be assumed, and what evidence would change the reader's mind. Its output is a set of design constraints rather than a description: each element of the analysis rules some design choices in and others out, and an artifact that cannot be traced back to the analysis is being designed against the analyst's own knowledge rather than the reader's.

Source: Adapted from Knaflic (2015) and Few (2006).

In other words, the reader is a specification in the sense Section 2.5 established, and audience analysis is what makes the specification writable. The five questions that do the most work are worth stating with StyleCraft's answers attached, because the answers are what dictate the design. Who is reading? A chief executive who is the sole decision-maker, with four colleagues in the room who have already read the pre-read and will not need convincing of the analysis, only of the recommendation. What decision? Whether to commit to eight leases, in whole, in part, or not at all, by Friday. How long? Ten minutes total, of which the first three seconds determine what she thinks she is looking at and the remaining minutes are spent testing that impression. What does she know? That StyleCraft opened seventeen stores in four waves; that the suburban customers spend more per order; and, because the VP's one-page summary said so, that the newest stores are young rather than weak. And what would change her mind? A comparable that failed, or a footprint figure that moved. What does she not know? Every number in the pre-read, the grain of any of them, and the vocabulary of Part II — she has never heard the phrase *prediction interval* used in a meeting she was chairing, and an artifact that requires her to learn it in the first minute has spent a tenth of its budget on vocabulary.

That last exchange is the one students find hardest to accept, and it is worth being precise about what it does and does not license. It does not license removing the uncertainty; Section 13.9 will insist on the opposite. It licenses expressing the uncertainty in a form that does not require a definition, which is a design problem rather than a compromise. It also does not license writing down to the reader. A chief executive who has run a retail business for a decade knows a great deal that the analyst does not, and the failure mode of oversimplification — a screen so reduced that it insults the reader's judgment and gives her nothing to test — is as real as the failure mode of density. What the analysis licenses is spending the reader's attention on the things only the analyst can supply.

The operational form of all of this is a test that takes three seconds to run and is the single most useful instrument in this chapter. It is a course heuristic rather than a standardized procedure, and it is worth saying so plainly, because the technique students will meet in the usability literature is normally a five-second first-impression test (Doncaster, 2014). Three seconds is a deliberately harder constraint, chosen because an executive dashboard on a projector in a room with an agenda gets less attention than a web page does, and because a test an artifact barely passes teaches less than one it must be designed to pass.

DEFINITION

The Three-Second Test (Course Heuristic)

The three-second test is the practice of showing a finished visual artifact to a representative reader for approximately three seconds, removing it, and asking what the artifact was about. Inspired by first-impression testing and by the glanceability principles of interface design, it operationalizes the fact that a reader forms a first impression within a few fixations and then interprets everything subsequent in light of it; the test therefore measures not whether the artifact can be understood but what it communicates before the reader has decided to work at it. This guide uses a three-second exposure as an intentionally demanding executive-dashboard constraint; the formal first-impression procedure in the usability literature is normally run at five seconds (Doncaster, 2014), and either duration is defensible provided it is stated.

The artifact passes when the reader's stated takeaway matches the analyst's intended takeaway, and the correct response to a failure is to revise the artifact, since the reader's response is the measurement rather than the error.

Source: Course heuristic developed for this guide, informed by Doncaster (2014), Few (2006), and Krug (2014).

In other words, three seconds is not a claim about how long executives read. It is an instrument for isolating what the artifact's visual hierarchy is doing, stripped of the reader's willingness to be helpful. Given thirty seconds, a cooperative colleague will find the finding in almost any dashboard, and the analyst will learn nothing about her design; given three, the colleague reports what the design actually pushed at her, which is the measurement of interest. Krug's (2014) argument about web pages transfers and is worth carrying: readers do not study an interface, they glance at it, form a plausible story, and act on the story, and a designer's job is to make the plausible story the true one.

Two rules of use keep the test honest. The first is that the analyst must predict the answer before running it, in writing, which is this chapter's version of the discipline the guide has enforced since Section 1.7 — a prediction converts the test from a reaction into evidence, and the gap between predicted and actual takeaway is the finding. The second is that the reader must be representative rather than convenient. A classmate who built the same dashboard last week is not a test of anything, because she knows where to look; the useful reader is someone who knows the business context and has not seen the artifact, which in a workplace means a colleague from another team and in this course means a peer working on different data.

Finally, a caution about what the test cannot do, since students routinely overextend it. The three-second test measures the top of the hierarchy and nothing else. It says whether the primary message lands; it says nothing about whether the supporting evidence is legible, whether the interactivity is discoverable, or whether the reader can find the answer to a question she thinks of in minute six. Those require the longer protocol of Section 13.10, which is a different instrument aimed at a different failure. An artifact that passes the three-second test and fails the ten-minute one is a poster rather than a dashboard, and the December 2 meeting needs both properties in the same object.

13.3 Exploratory vs. Explanatory Artifacts

Section 13.2 specified the reader. This section takes the next decision, which comes before layout and before any chart is drawn, and which the first draft skipped entirely: what kind of object is being built. Chapter 12 distinguished visualization's two jobs and assigned this chapter the second one, but "communication artifact" is not a form. It is a category containing at least three forms with different structures, different reading protocols, and different failure modes, and choosing among them is the highest-leverage decision in the chapter, because a well-built object of the wrong kind cannot be repaired by refinement.

DEFINITION

Exploratory and Explanatory Artifacts

An exploratory artifact is designed to let a reader investigate a body of data by her own path, supporting questions the builder did not anticipate; its structure is a space to be navigated, and its success criterion is coverage of the questions a reader might legitimately ask. An explanatory artifact is designed to deliver a specific finding along a path the builder chose, supporting a question the builder has already answered; its structure is a sequence, and its success criterion is that the intended understanding is acquired in the time available.

The distinction is not the same as the analysis-communication distinction of Section 12.2: an artifact may be built for an external audience and still be exploratory, and the most common design error in business visualization is building an exploratory object for a reader who needed an explanatory one.

Source: Adapted from Knaflic (2015) and Segel and Heer (2010).

In other words, the question is whether the reader is looking for something or being shown something, and it is answered by the decision rather than by the data. Three forms follow, and a fourth object travels with all of them; Table 13.1 places them against the readers and decisions they serve.

Table 13.1

Choosing the Artifact from the Decision and the Reader

Artifact	What it is	Reader and time	Use when	Fails when
Dashboard	A single screen of coordinated views, usually refreshing, with a fixed set of questions designed in	A recurring reader with minutes, returning on a schedule	A decision or a monitoring task recurs on a known cadence and the same questions are asked each time	The decision is one-time; the reader is new to the material; the questions cannot be anticipated
Report	A linear document combining prose, charts, and tables, read at the reader's pace	A reader with an hour, reading alone, who may need to re-read	The evidence is long, the reasoning must be inspectable, and the reader will need the document again later	The reader will not read it; the decision happens in a meeting nobody prepared for
Data story	An ordered sequence of views with narrative connective tissue, delivered live or as a designed scroll	A captive audience with a fixed span, in a room or on a screen	A specific finding must be transferred once, in order, to people who have not seen the evidence	The reader needs to explore; the finding is genuinely contingent on the reader's own question
Recommendation memo	Prose stating the decision, the evidence, the uncertainty, and the conditions for reversal	A decision-maker, before or after the meeting	A decision requires a record of what was recommended and on what basis	Substituted for the visual evidence rather than paired with it

The pre-read of Chapter 12 was, in these terms, an exploratory artifact for an external audience — deliberately, because its readers' job was to find its weaknesses. The December 2 artifact is explanatory and, on the surface, a dashboard. The interesting part is that a strict reading of Table 13.1 says a dashboard is the wrong form here, because the decision is one-time and the reader is new to the material, and the analyst should notice this rather than shrug at it. What resolves the conflict is that the artifact is not being handed to the chief executive to read alone; it is

being presented, live, in a room, by the person who built it, with a memo attached. That makes it a data story delivered on a dashboard's canvas, with the dashboard's interactivity reserved for the questions the room asks after the story lands — and it is the reason Lab 13.1 builds a single screen with a strict reading order rather than the free-navigation grid the first draft produced. Naming this explicitly is not pedantry. It settles half a dozen later decisions: it is why the filter panel disappears entirely, why the views are unequal in size, why the titles carry sentences, and why there is a memo at all.

DEFINITION

Data Story

A data story is an ordered sequence of views, each carrying a stated point, connected so that the sequence itself constitutes an argument — typically moving from the situation the reader already accepts, through the complication the evidence introduces, to the resolution the analyst recommends. Its defining property is that the order is chosen by the author and is load-bearing: rearranging the views changes what the audience concludes, which is what distinguishes a data story from a collection of correct charts about the same subject.

Source: Adapted from Segel and Heer (2010) and Knaflic (2015).

One further note closes the section, and it is the one the Business Analytics in Practice section will find running through every organization that has deployed self-service tools at scale. The dashboard is over-selected. It is what stakeholders ask for by name, it is what tool vendors sell, and it is what analysts default to because it looks like the most work for the least commitment — a dashboard implies no particular conclusion, which is comfortable. The discipline this chapter asks for is to answer the question in Table 13.1's *Use when* column honestly before opening a canvas. A one-time decision made in a meeting by a reader who has never seen the data is a data story with a memo attached, and building it as a dashboard because a dashboard was requested is how an analyst spends two days producing something nobody opens twice.

13.4 KPI Hierarchy on a Dashboard

Section 13.3 settled what kind of object is being built. This section supplies its internal structure, and the structure is the direct spatial expression of a discipline students have owned since Chapter 3. Section 3.5 distinguished metrics from KPIs, and the distinction was that a KPI is a metric elevated by a decision: of the dozens of quantities an analysis can produce, a small number are the ones a specific choice actually turns on. That elevation has been carried in prose ever since. On a dashboard it has to be carried in space, because space is what a reader decodes first and fastest, and a canvas on which every view is the same size has flattened the hierarchy and thereby thrown away the analyst's most important judgment.

DEFINITION

KPI Hierarchy

A KPI hierarchy is the ordering of a deliverable's metrics into tiers by their proximity to the decision — a primary tier of one or two measures on which the decision turns, a supporting tier that explains or qualifies the primary, and a diagnostic tier that answers the follow-up questions the primary provokes — and the expression of that ordering in the artifact's visual prominence, so that size, position, and emphasis correspond to decision relevance. Because readers allocate attention by prominence before they read any label, a hierarchy that is stated in the analyst's head but not in the layout is not communicated at all.

Source: Adapted from Few (2006) and the metric and KPI roles of Section 3.5.

In other words, the first design act on a canvas is to decide what the reader should look at first, second, and third, and the second design act is to make the canvas say so. StyleCraft's case makes the tiers concrete, and building them requires going back to the decision rather than to the pre-read. The decision is whether eight leases are signed. The quantity that decision turns on is not revenue, is not average order value, and is not the repeat-order share that dominated the pre-read's argument; it is how long a new store takes to reach, on a four-week average rather than in a single week, the point where its contribution covers its own fixed footprint. Everything else on the screen is there to explain that number, to qualify it, or to answer the objection it invites. Table 13.2 sets the tiers out in the order the reader will meet them, and it is written to be filled in before a canvas is opened rather than read once.

Table 13.2

The KPI Hierarchy of the Go/No-Go Dashboard

Tier	What it must establish	The StyleCraft measure	Visual treatment	Reader's question it answers
Primary	Whether a store clears its own fixed cost, and when	First four-week-average coverage: the first store-age week in which the four-week average of weekly contribution reaches the $1,769 that carries the $92,000 annual fixed footprint, for each mature suburban and resort store	Largest object on the canvas, upper left, with the threshold drawn, the four-week-average coverage week labeled on each of the four lines, and the earlier one-week crossing marked more faintly	"Do these stores cover their own cost, and when?"
Primary	Where the eight new stores sit against that path	Wave 4 store ramps overlaid on the range their predecessors occupied at the same store age	Second-largest, adjacent to the primary, sharing its vertical scale	"And are the new ones on track?"
Supporting	Why the suburban economics differ at all	Average order value by metro, computed at order grain, with the chain average drawn	Half-height, beneath the primaries, first in reading order	"What is actually different about these customers?"
Supporting	Whether that difference is basket composition or price	Category revenue share by metro, with occasionwear identifiable	Half-height, second in the supporting band	"Is it a mix story or a pricing story?"
Supporting	What the planning year is worth, and how much of it is still open	The FY2027 chain contribution outlook — actual through November, forecast thereafter — with its 80 and 95 percent regions on the forecast months and a labeled horizon-total block, annotated with the open range in store-equivalents	Half-height, rightmost in the supporting band	"How much of this is a guess?"
Diagnostic	The known weakness in the case	Repeat-order share by store type against the chain share, with the suburban gap labeled	Small, lower band, de-emphasized but present	"What are you not telling me?"
Diagnostic	The store that does not fit the pattern	The one Wave 4 store tracking below the predecessor range, named, with its ramp	Small, lower band, adjacent to the above	"Which sites would you not sign?"

Three properties of that table deserve comment, because each one is a decision students tend to skip. The first is that the primary tier contains two views rather than one, and that this is close to the limit. A hierarchy with four

primaries has no primary, and the discipline of forcing the count down to one or two is where the analyst discovers what she actually believes the decision turns on. The second is that the diagnostic tier contains the case's two weakest facts — the suburban repeat-order share of 24 percent against a chain share of 47 percent, and the Scottsdale store that has tracked below the predecessor range since week six — and that they are on the screen rather than in a backup slide. Putting them in the diagnostic tier is not a concession; it is what makes the primary tier believable, and Section 13.15 will argue that it is also the ethical requirement. An executive who discovers a weakness herself concludes that the analyst either missed it or hid it, and both conclusions are expensive. The third is that every tier is defined by a question the reader will ask, in the order she will ask it, which is what turns a hierarchy into a sequence and hands the next section its material.

One boundary belongs here. The tiers govern prominence, not inclusion, and the temptation the tier structure creates is to solve a crowded canvas by demoting rather than by cutting. A view that has no tier does not become a small view; it comes off the screen and goes into the appendix of the memo, where it remains available to anyone who asks and stops competing for the ten minutes. Section 13.7 gives the general form of that rule, and it is the hardest habit in this chapter to acquire, because everything on the first draft was work the analyst did.

13.5 Dashboard Layout

The hierarchy of Section 13.4 is a claim about importance. This section makes the claim visible, and it does so through four instruments — reading order, grouping, density, and whitespace — that together convert a list of tiers into a canvas a reader traverses in the intended sequence. The section is deliberately practical, because layout is where the previous three sections either become a design or remain an intention.

Begin with reading order, which is the instrument that carries the sequence. Readers of left-to-right languages enter a rectangular canvas at the upper left and move rightward and downward, and the eye's path across a screen is shaped by that habit and by the size and contrast of what it encounters — a large, high-contrast object anywhere on the canvas will capture the entry fixation regardless of position, which means position and prominence must agree or they will fight. The practical rule that follows is simple and is violated constantly: place the primary tier at the upper left and make it the largest object, place the supporting tier along the path the eye takes next, and place the diagnostic tier last in that path. The go/no-go dashboard therefore runs the coverage view across roughly the upper-left 40 percent of the width, the Wave 4 ramp comparison immediately to its right at about 35 percent and sharing its vertical scale, the three supporting views as a half-height band beneath them, and the two diagnostics as a smaller band at the bottom. That arrangement is not a grid; it is a sentence, and its reading order is the argument of Section 13.6 rendered as geometry.

Grouping is the second instrument, and it works by proximity and enclosure. Objects placed close together are read as related and objects separated by whitespace or a rule are read as distinct, and this happens pre-consciously and cannot be overridden by a label. The two consequences that matter on a dashboard are that views belonging to the same tier should be visually adjacent and share their formatting, and that views sharing a scale or an axis should be aligned so that the shared scale is visible as such. The Wave 4 ramp view sits directly beside the cost-coverage view for exactly this reason: the two carry the same measure on the same vertical range, and placing them adjacent with aligned axes lets the reader compare across identical non-aligned scales — one rung down the perceptual hierarchy of Section 12.4 rather than several, which is the best available given that they cannot share a single frame.

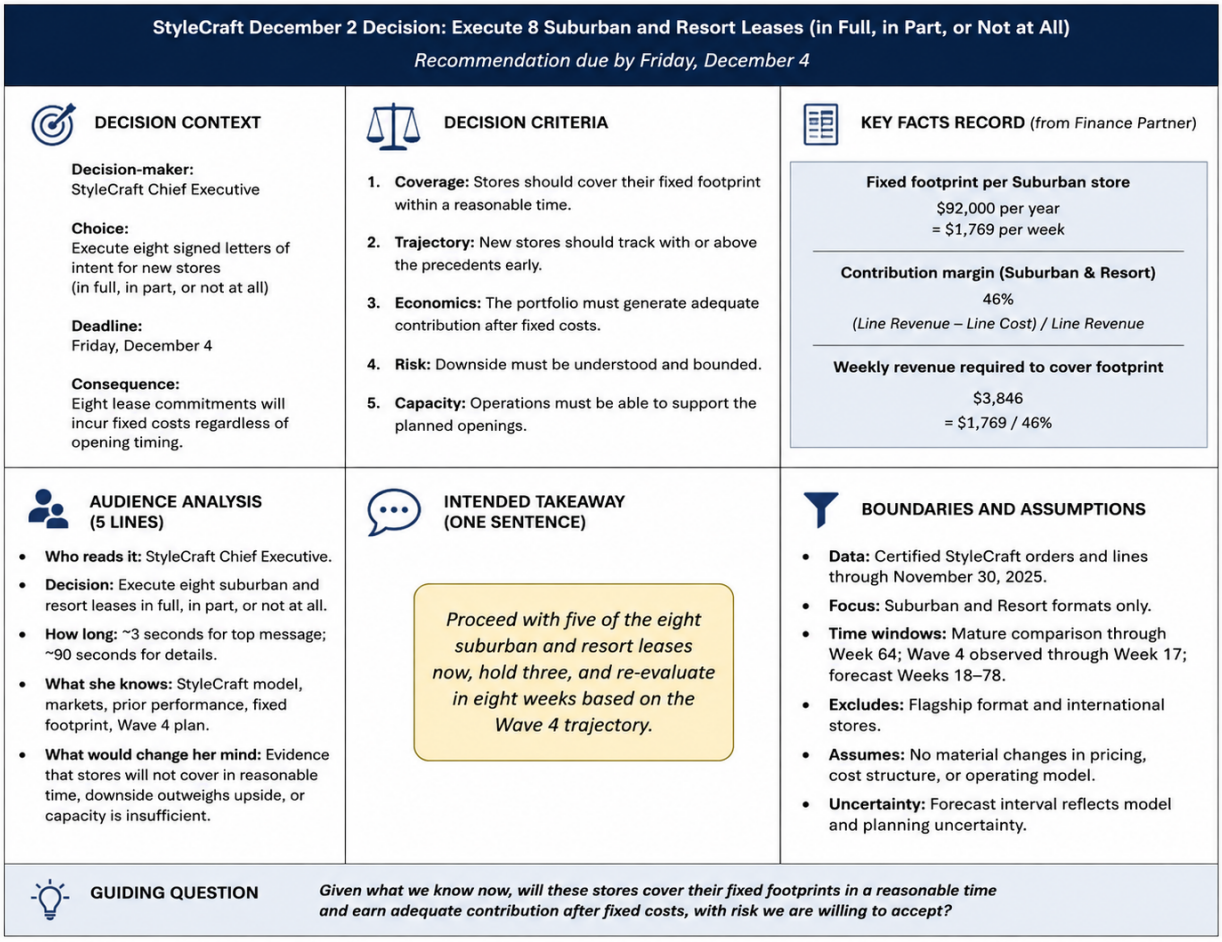
Density is the third and is where executive dashboards most often fail in the direction opposite to the one students expect. The failure is not too few views; it is too many marks per view and too many views per screen, each individually defensible. Miller's (1956) observation that people hold only a handful of items in working memory is invoked constantly in design writing and is often overinterpreted, but the practical form of it survives the caveats: a reader who must hold eight findings simultaneously in order to reach a conclusion will not reach it, and a canvas of eight equal views demands exactly that. The counter-discipline is to state, before laying anything out, the number of things the reader must hold at once — for the go/no-go dashboard it is two, the cost-coverage path and the new stores' position on it — and to build the layout so that everything else is available in sequence rather than simultaneously.

Whitespace is the fourth, and it is the one students cut first and should cut last. Empty space is not wasted space; it is the mechanism by which grouping and hierarchy are expressed, and a canvas with generous margins around its primary view is telling the reader where to look as surely as an arrow would. The most common mistake in Tableau specifically is to leave the sizing decision to the software without deciding what consumption condition the artifact is built for. Tableau offers Fixed size, Range, Automatic, and device-specific layouts, and each is correct for a different situation (Tableau, 2026e). Automatic stretches every object to consume whatever window it is opened in, which is the right behavior for a dashboard published to the web and opened on unknown displays and the wrong behavior for an artifact whose hierarchy is expressed in relative area and which will be projected at a known resolution in a known room. The go/no-go dashboard is built Fixed at the projection resolution because the room is known; a monitoring dashboard read on phones by eight field managers should use device-specific layouts instead, and the choice between them is a statement about where the artifact will be read rather than a preference.

Figure 13.1

The Go/No-Go Dashboard, as a Wireframe

Figure 13.1 The Decision Before the Dashboard



CONCEPT

Position Is a Claim About Importance

Every element on a canvas occupies a location, and the reader decodes location before she decodes anything else — before the title, before the axis, before the numbers. This means an analyst who has not decided where things go has still made a claim about what matters, and has made it by accident, usually in the order the views happened to be built. The habit this section asks for is to treat placement as an assertion that must be justified in the same way a title must: for every view on the canvas, be able to state why it sits where it sits, in terms of the tier it belongs to and the point in the reader's sequence at which she needs it. Two consequences follow immediately, and both are uncomfortable.

The first is that the upper-left position is scarce and cannot be given to the view the analyst is proudest of; it belongs to the view the decision turns on, which is frequently a plainer object. The second is that a view for which no justification can be stated does not belong on the canvas at all — and the honest test of whether the hierarchy is real is whether anything was removed to make room for it.

Two structural mechanics finish the section, since Tableau supplies them and the labs will use them. Layout containers — horizontal and vertical — are the mechanism by which the tiers are actually held in place, and building a dashboard out of nested containers rather than floating objects is what makes the layout survive a change in screen size or a late addition. And a dashboard needs a title band and a source band, which are the communicative descendants of Section 12.9's footer discipline: the title band carries the decision the screen supports and the date, and the source band carries the extract, the window, and the filter state in small type. What changes from Chapter 12 is not whether the provenance appears but how much of it appears at each level. Chapter 12 put five lines under every view because every view was going to be interrogated individually; a dashboard carries the provenance once, for the screen, with per-view grain statements moved into tooltips where the reader who wants them can find them and the reader who does not is not taxed. That is a legitimate compression and not a concealment, and the line between the two is exactly what Section 13.15 has to define.

13.6 Storytelling with Data

Section 13.5 arranged the views in the order the reader will traverse them. This section supplies the reason the order is that order, and the sentences that make each stop on the path mean something. The word *storytelling* has been abused enough in analytics marketing to make graduate students suspicious of it, and the suspicion is healthy, because a great deal of what travels under the name is decoration applied to a deck. What the term denotes here is narrow and technical: an argument with a stated order, in which each view carries one claim, and in which the sequence is doing work that no individual view does.

The structure that has proven most durable is three-part and predates data visualization by a considerable margin. It opens with the situation, which is the state of affairs the audience already accepts and which exists to establish common ground rather than to inform; it introduces the complication, which is the fact that makes the situation inadequate and which is where the analysis enters; and it closes with the resolution, which is what the analyst recommends doing about it. StyleCraft's version writes itself once the KPI hierarchy is fixed. The situation is that StyleCraft opened seventeen stores in four waves and that the suburban customers spend more per order than the urban ones — which everyone in the room believes and which the composition and order-value views establish in ten seconds. The complication is that revenue per order is not the quantity the lease decision turns on; the quantity is the time it takes a store to reach, on a four-week average rather than in a single week, the weekly contribution that covers its own fixed footprint, and on that measure the mature stores' four-week moving averages did not first reach that threshold until somewhere between weeks 43 and 57, which is longer than anyone in the room has been assuming. The resolution is that the eight sites are not one decision but three: the ones whose matched comparable

reached four-week-average coverage, the one whose matched comparable did not, and the ones with no qualifying comparable at all.

Notice what the sequence accomplishes that no single view does. It converts the finance partner's question from an objection into the organizing question of the artifact, which is generally the right move when a skeptical reader has already voiced one — a story built around the strongest objection is far more persuasive than one built around the strongest evidence, because the audience is not left holding an unaddressed doubt while the analyst talks. Segel and Heer (2010) surveyed narrative visualization across journalism and analysis and identified this same tension as the genre's organizing axis: every such artifact sits somewhere between author-driven ordering, which is efficient and directive, and reader-driven exploration, which is thorough and slow. The go/no-go dashboard sits well toward the author-driven end for its first minute and moves toward the reader-driven end afterward, which is a design position rather than a compromise, and it is implemented through the interactivity of Section 13.8.

The sentence-level instrument is the title, and it is where most of the chapter's persuasive power and most of its ethical risk are concentrated.

DEFINITION

Assertion Title

An assertion title is a chart title that states the finding the view supports, written as a complete sentence with a subject and a verb — "The four mature suburban and resort stores' four-week moving averages first reached the weekly fixed-cost threshold in weeks 43, 46, 51, and 57" — in contrast to a label title, which names the fields displayed — "Weeks to Weekly Coverage by Store." Because a reader acquires the title before decoding the marks and interprets the marks in light of it, an assertion title determines what the view is understood to show; the technique therefore carries an obligation that a label title does not, which is that the sentence must be verifiable against the view directly beneath it, in the view's own units, without additional evidence.

Source: Adapted from Knaflic (2015) and Kosslyn (2006).

In other words, the assertion title is the insight statement that Chapter 5 distinguished from a mere observation, written above the view that proves it — which is why it is the single highest-return technique in this chapter and the single most dangerous, and both facts have the same cause: the reader believes it. Three rules keep it honest, and they are what Lab 13.2 checks. The first is that the sentence must be true of the view and not merely true — a title reading "Suburban stores are our best growth opportunity" may be defensible somewhere in the analysis and is not defensible above a chart of weeks-to-coverage, and a reader who verifies the sentence against the marks will find that it cannot be done. The second is that the verb must be one the evidence licenses, which is Section 11.10's evidence hierarchy arriving in the title bar: *rose*, *is*, *reached*, and *differs* are descriptive verbs a view can support; *drives*, *causes*, *proves*, and *will* are claims about mechanism or the future that a descriptive view cannot carry, however strong the underlying analysis. The third is that a number in the title must be a number in the view. Titles that assert a quantity the reader cannot find in the marks — "Suburban expansion adds $1.2 million" over a chart showing weeks — are the most common way a fluent title outruns its evidence, and they are especially common when the title is drafted by an assistant that has been given the chapter's conclusion but not the view's data.

Table 13.3 works the transformation for the go/no-go dashboard's principal views, and the exercise of writing all three columns is itself the discipline: the label title is what the analyst has; the assertion title is what she believes; and the *What must be visible* column is the check that stops the second from drifting past the first.

Table 13.3

Label Titles and Assertion Titles for the Dashboard's Principal Views

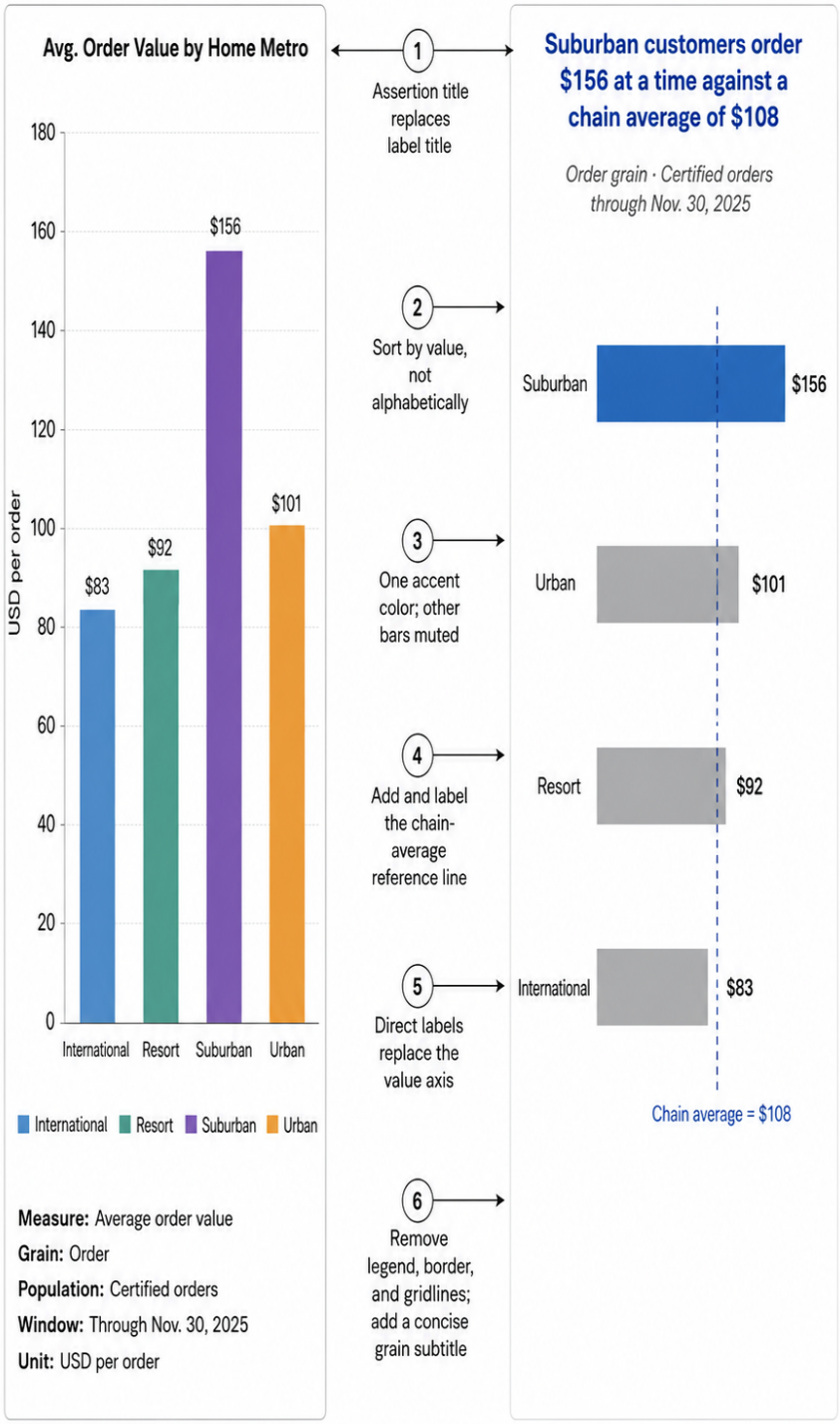
View	Label title (what it plots)	Assertion title (what it shows)	What must be visible in the view for the assertion to be earned
Coverage path	Contribution 4wk Avg by Store Age and Store	"The four mature suburban and resort stores' four-week moving averages first reached the weekly fixed-cost threshold in weeks 43, 46, 51, and 57"	All four stores drawn as four-week averages, the $1,769 weekly threshold drawn, the four-week-average coverage week labeled on each line, the earlier one-week crossing marked, and the zero-order weeks present rather than skipped
Wave 4 ramps	Contribution 4wk Avg by Store Age and Store	"Seven of the eight new stores are within or above the range their predecessors occupied at the same age"	The predecessor range as a region whose edges move week by week; exactly eight lines in the frame and no others, all of them Wave 4 and all suburban or resort; a shared vertical scale; store age on the horizontal axis
Order value	Average Order Value by Home Metro	"Suburban customers order $156 at a time against a chain average of $108"	Both figures as marks, the chain average drawn as a reference line, order grain stated
Mix	Category Revenue Share by Metro	"Occasionwear is roughly three times the share of suburban revenue that it is of urban"	Both metros' shares, occasionwear identifiable, shares summing to 100 percent
FY outlook	Monthly Chain Contribution with Prediction Intervals	"The FY2027 outlook is $1.25 million of chain contribution, and the range still open on the remaining seven months spans about 1.8 mature stores' annual contribution end to end"	The monthly line, the seam line where reporting ends with no shading to its left, and both interval regions to its right; a labeled horizon-total block splitting actual from forecast, since neither total is readable from the monthly marks; and the store-equivalent annotation
Repeat share	Repeat-Order Share by Store Type	"Suburban repeat-order share is 24 percent against a chain average of 47 percent"	Both figures, the chain reference line, store types sorted

Two general points close the section. The first is that the assertion title changes what a chart is for, and therefore what must be on it. The moment a title says "seven of the eight," the view acquires an obligation to show eight things distinguishably, which the pre-read's version did not, and the title has consequently forced a design change rather than merely labeling one — this is the productive direction of the technique and the reason titles should be drafted early rather than added at the end. It also disciplines the wording: the same view supports "within or above the predecessor range" and does not support "at or above the band," which reads as though all seven sat at the upper boundary or beyond it when in fact two are above the range and five are inside it. The second is that the title is where the analyst's voice is most exposed and should therefore be the last thing checked before the artifact ships. Section 13.11 makes that check a routine, and Lab 13.2 makes a peer run it, because the author of a sentence is the worst available judge of whether it says more than she can prove.

Figure 13.2

The Same Data as an Analytical View and as an Explanatory View

Figure 13.2 The Same Data as an Analytical View and as an Explanatory View



13.7 Emphasis, De-Emphasis, and Decluttering

Section 13.6 gave each view a sentence. This section makes the sentence findable, and it does so with two operations that are usually taught separately and are in fact one operation performed in two directions. Emphasis makes an element stand out. Decluttering removes the elements it would otherwise have to stand out against. Neither works alone: emphasis applied to a busy canvas produces a busy canvas with one loud thing in it, and decluttering without emphasis produces a clean canvas with no entry point.

The mechanism underneath emphasis is perceptual and has been measured, which lets this section do for emphasis what Section 12.4 did for accuracy.

DEFINITION

Preattentive Attributes

Preattentive attributes are visual properties that the visual system processes in parallel across much of the visual field, before focused attention is directed anywhere, so that an object differing from its neighbors in one such property can be detected rapidly and with little increase in search time as the number of neighbors grows. The principal attributes available in business charts are color hue, color intensity, size, position, orientation, shape, enclosure, and added marks. The effect is strongest under favorable conditions — a single distinctive feature, homogeneous distractors, adequate contrast — and it degrades with target-distractor similarity, with heterogeneous displays, when the target is defined by a conjunction of two attributes, and when several attributes vary at once.

It is therefore a design resource with conditions attached rather than a guarantee of constant-time detection.

Source: Adapted from Treisman and Gelade (1980) and Healey and Enns (2012).

In other words, a reader finds a red bar among gray bars without searching, finds a large mark among small ones without searching, and does search — slowly, and with errors — for the bar that is both red and large in a field where some bars are red and some are large. Treisman and Gelade's (1980) feature-integration experiments established the pattern and the conjunction penalty; Healey and Enns (2012) reviewed four decades of the subsequent work in visualization terms, including the ways the effect weakens as displays become less obliging. The practical consequence for a dashboard is a budget rather than a technique: preattentive emphasis works because it is scarce, and a canvas that emphasizes six things has emphasized nothing, because the reader must now search the emphasized set. Table 13.4 lists the attributes with the use each is suited to and the failure each produces when overspent.

Table 13.4

The Emphasis Toolkit

Attribute	What it does well	Use on the go/no-go dashboard	Failure when overspent
Color hue	Marks one category as the subject; separates a small number of groups	The four mature suburban and resort stores in the accent hue; every other store gray	Every series in a different hue, so the reader must consult a legend to know what she is looking at
Color intensity	Orders a quantity; recedes a background element	The predecessor ribbon as a pale fill behind the Wave 4 lines	Used for an unordered field, asserting an order (see Section 12.8)
Size	Signals importance at the layout level; carries a rough magnitude	The primary tier's views drawn largest on the canvas	Size differences within a view read as magnitude claims that were not intended

Attribute	What it does well	Use on the go/no-go dashboard	Failure when overspent
Position	Carries importance in reading order; carries value precisely	Primary tier upper left; supporting tier beneath	Prominence and position disagree, so the eye is pulled away from the intended entry point
Enclosure	Groups elements as belonging together; isolates a region of a chart	A light box around the eight Wave 4 lines in the ramp view	Boxes around everything, which restores the undifferentiated grid
Added marks	Directs attention to specific places, and loses its force as their number grows	One labeled annotation at each store's four-week-average coverage week on the coverage view — four in all, and nothing else annotated	Annotations on every mark, which become a second layer of clutter
Direct labels	Removes the legend lookup entirely	Store names at the end of the four mature lines	Labels on all seventeen lines, which is a thicket

The second operation is subtraction, and its governing idea is Tufte's.

DEFINITION

Data-Ink and Decluttering

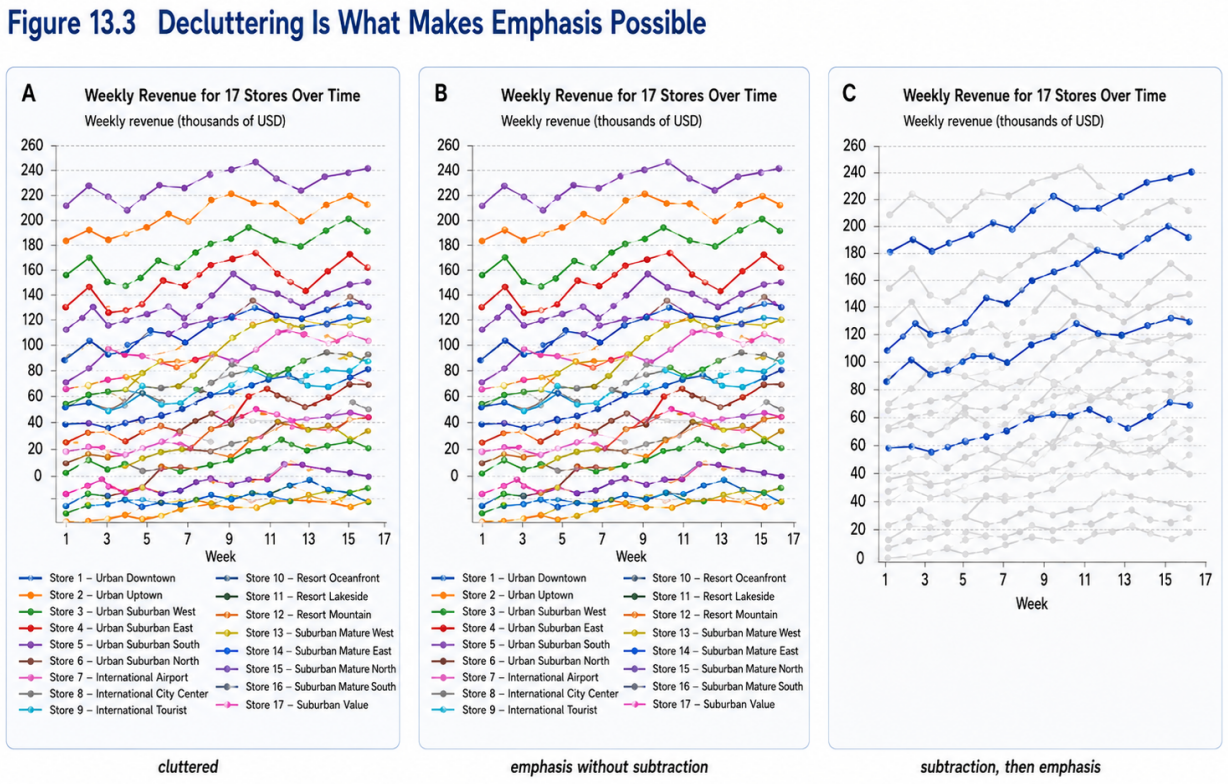
Data-ink is the portion of a chart's ink that represents data values and would be lost if the data changed; non-data-ink is everything else — gridlines, borders, backgrounds, tick marks, legends, redundant labels, and decoration. Decluttering is the systematic removal or de-emphasis of non-data-ink and of duplicated data-ink, undertaken not for austerity but to raise the contrast between what carries meaning and what does not, so that emphasis has something to work against. The discipline is a ratio rather than a minimum: elements that help a reader interpret correctly, including reference lines, direct labels, and annotation, are retained even though they encode no data value.

Source: Adapted from Tufte (2001) and Few (2012).

In other words, the rule is not "remove everything" but "remove everything that is not doing a job you can name," and the qualification matters because the naive version of data-ink maximization produces unreadable charts. Section 12.9's reference lines are non-data-ink and are mandatory; the footer is non-data-ink and is an ethical instrument; a gridline that lets a reader read a value off a long axis is non-data-ink doing real work. What comes off the go/no-go dashboard is the material the first draft inherited from the analytical version and never questioned: the heavy borders around every worksheet, the gray background panels, the tick marks at every increment, the axis titles that repeat what the assertion title already says, the four legends replaced by direct labels, the three views whose findings the memo carries better than a chart does, and all seven filter controls. That last removal is the largest single improvement to the draft, and it is a removal of function rather than of decoration, which is why Section 13.8 has to justify it.

Figure 13.3

Decluttering Is What Makes Emphasis Possible



CONCEPT

Decluttering Is Deleting Your Own Work

The reason decluttering is hard is not that analysts lack taste. It is that every element on a first draft is there because somebody put it there, usually for a reason that was good at the time, and removing it feels like discarding effort — which it is. The two habits that make it survivable are procedural rather than aesthetic. The first is to declutter against the artifact's stated sentence rather than against a general principle: with the assertion title of Section 13.6 written down, every element on the canvas can be asked whether it helps a reader believe that specific sentence, and elements that do not have a defensible answer come off without further argument. The second is to move rather than to delete, at least at first.

A view that does not earn its place on the screen goes into the memo's appendix; a filter that does not earn a control goes into the workbook the analyst brings to the room; a footnote that does not earn space in the source band goes into the tooltip. Nothing is lost, the reader's ten minutes are recovered, and the analyst is spared the psychological cost that makes this step the one most often skipped. What is not permitted is the third option, which is to shrink everything until it fits — a canvas of twelve small views is not a decluttered canvas of twelve views, it is an illegible one.

13.8 Interactivity as Navigation and Communication

Sections 13.6 and 13.7 built an artifact that says one thing clearly. This section takes up the property that makes a dashboard different from a printed page, and it argues for a use of that property considerably narrower than the tool invites. Interactivity in this chapter is a means of navigation and of communication: it lets a reader move through an argument at her own pace, ask a follow-up the analyst anticipated, and see a detail without cluttering the canvas. It is not, in this chapter, a means of computation. Parameters that recompute a metric, what-if controls, threshold sliders that change what a view is measuring, and set actions that build cohorts on the fly are analytic instruments rather than communicative ones; they belong to the advanced visual-analytics work that follows this guide, and a student who reaches for them here should notice that she is rebuilding the exploratory artifact Section 13.3 already ruled out.

The organizing idea is Shneiderman's (1996) mantra, which has aged well: overview first, zoom and filter, then details on demand. Read as a design sequence rather than as a slogan, it says that the reader should be able to see the whole before she is asked to choose a part, that filtering should narrow a view she already understands rather than construct one she does not, and that detail should arrive when requested rather than occupy the canvas permanently. Heer and Shneiderman (2012) later cataloged the interaction types that implement it, and the four that matter for a marketing dashboard are worth naming with their communicative jobs attached. A filter narrows the population under discussion and answers "what about just the suburban stores?" A highlight action marks the same entity across several views at once and answers "where else does Scottsdale appear?" A tooltip supplies the row-level or grain-level detail the canvas cannot carry and answers "what exactly is this mark?" And a navigation action moves the reader from one screen to another and answers "show me the evidence behind that." Each of those is a question the analyst predicted; that is the test of whether the interaction belongs, and it is what the definition below turns into a design obligation.

DEFINITION

Dashboard Action

A dashboard action is an interaction configured on a dashboard by which activity in one view — a hover, a selection, or a menu choice — changes another view, by filtering it, by highlighting a matching entity within it, or by navigating the reader to a different sheet or an external resource. A highlight action operates on matching field values, so it can only mark an entity in a target view that actually contains that entity; a highlight configured from a source view onto a target whose data does not include the selected value does nothing, silently.

An action's communicative purpose is to answer a follow-up question the builder anticipated without giving that question permanent space on the canvas; its design test is therefore that the analyst can name the reader's question the action answers, that the target views contain the values the action will pass them, and that a reader who never triggers it still acquires every finding the artifact must communicate.

Source: Adapted from Heer and Shneiderman (2012), Shneiderman (1996), and Tableau (2026b).

That second condition is where most configured actions quietly fail, and the go/no-go dashboard supplies two examples of it, one of which is a direction rather than a target. A highlight on `store_name` cannot "trace one store across the argument" into the cost-coverage view at all, because that view contains only the four mature suburban and resort stores and shares no store name with Wave 4; it is left out of the action rather than included and inert. The subtler point is which of the two remaining views is the source. Sourcing from the eight-store ramp view onto the single-store outlier view fires for one selection in eight and does nothing for the other seven — not an error message, just no response, which a reader interprets as a broken dashboard rather than as an empty match. Sourcing from the outlier view onto the ramp view fires for every selection it can produce, because the one store it contains is one of the eight the target holds. The general rule is that an action should run from the smaller value set to the larger one, and Lab 13.1 Part C builds it that way. Table 13.5 states each interaction's communicative case, its concealing

case, and the disclosure it requires, and its *Conceals when* column is the one to read carefully, because every entry in it describes something an analyst does by accident rather than by design.

Table 13.5

Interactivity That Communicates and Interactivity That Conceals

Interaction	Communicates when	Conceals when	The disclosure it requires
Filter control	It narrows a population the reader already understands, the default state is the honest one, and the set of worksheets it applies to is stated	The default is a restricted state the reader assumes is complete; the control is the largest object on the screen; it silently changes a view whose title asserts a chain-wide figure	The current filter state and the worksheets affected, visible as text on the canvas, not only in the control
Highlight action	It links the same entity across views that all contain that entity, so a reader can trace one store through the part of the argument where it appears	It is the only way to see a comparison the reader needs, so a passive reader never sees it; or its target views do not contain the field values, so it appears to do nothing	Nothing; but the comparison must also be legible without the interaction, and the action must run from the smaller value set to the larger so that every selection it can produce has a match
Tooltip	It carries grain, definition, and provenance for readers who want them	It carries the qualification that makes the headline defensible	Anything load-bearing moves out of the tooltip and onto the canvas
Navigation action	It moves a reader from a summary to its supporting evidence on request	It buries a finding two clicks deep so the summary reads cleaner than the evidence supports	A visible cue that the deeper view exists
Parameter used as navigation	It swaps which of several equivalent views is displayed — one metro at a time, say	It changes the evidence on display in a way the title does not track	The title must be built dynamically so that it remains true in every state the control can produce

The last row is the one that produces real accidents, and it is worth stating as a rule because it is the intersection of this section with Section 13.6. An assertion title is a sentence about a view; a parameter changes the view; therefore an assertion title on a parameterized view must change with it, or the analyst has shipped a screen that is true in one state and false in the others. Tableau supports inserting a field or parameter value into a worksheet title, so the sentence can be assembled in a calculated field and inserted (Tableau, 2026f); Lab 13.1 Part C does it, and the failure is common enough in circulated workbooks that it is worth looking for in anyone else's.

Filters carry a quieter version of the same problem, and it is worth being explicit because Tableau's scoping options make it easy to get wrong. A filter applied to all worksheets using the same data source will change every view built on that source, including views whose titles assert chain-wide quantities and views whose reference lines were computed on a different population (Tableau, 2026c). Scoping is the first tool to reach for: a store-type filter belongs on the views that are about physical store performance and not on the order-value, mix, outlook, or repeat-share views, whose assertion titles quote chain-wide figures that must not move when the reader narrows the store set. Reference values entered as constants from a certified metrics file stay fixed under filtering by construction; a reference computed from the view changes with the marks, so filtering it moves the benchmark as well as the bars and the title becomes false. But scoping is not always sufficient, and the next paragraph is the case where it is not.

Sometimes no scope is safe, and then the control comes off. That is what happens to both of the go/no-go dashboard's surviving filters, and the two cases fail differently, which is why both are worth following. A date-window control fails on the *titles*: the order-value, mix, and repeat-share views quote full-window figures in their sentences and draw full-window constants as reference lines, so narrowing the window moves the marks and leaves four numbers and two reference lines standing behind them, unchanged and now wrong. A `store_type` control fails on the *rows*. Scoped to the two ramp views and the outlier view it can empty the coverage view entirely when the reader deselects a format; it can change the number of lines in a frame whose title says "seven of the eight"; and

— because the predecessor region's outline rows carry no store type at all — it can delete the region while leaving the subtitle that describes it. Scoping narrows which worksheets a filter touches; it does not keep a sentence true, and it does not protect rows for which the filtered field is null (Tableau, 2026c).

Lab 13.1 Part C therefore ships an executive artifact with **no filter controls on it**, and states the reasoning rather than leaving it to be inferred. The two alternatives were to make six titles dynamic — spending the reader's ten minutes on questions she has not asked — or to remove the controls and declare the state in words instead. A screen with no controls makes no claim about its own completeness, so the source band has to carry the population, the window, the exclusions, and the sentence *no filters applied*. All seven controls live on an analytical tab in the same workbook, one click away for the analyst and invisible to the room. The general rule this produces is worth carrying: a control that can falsify a sentence on the same screen is not a convenience, and "the reader can always set it back" is not an answer, because the reader does not know what it was set to.

Two cautions close the section. The first is that interactivity is invisible until it is discovered, and readers in meetings do not explore. An executive who has ten minutes will click nothing; a colleague who opens the workbook on Thursday will click a little; the analyst who built it will click everything and will therefore badly overestimate how much of her design the audience will ever see. The working rule is that every finding the artifact must communicate has to be visible in the default state, and interactivity may only add depth to a finding that has already landed. The second is the one the first draft violated: the presence of many controls is itself a message, and the message is "this is a tool for you to use," which is the wrong message for a ten-minute reader making a one-time decision. Cutting the go/no-go dashboard's filter panel from seven controls to none is a communication decision rather than a simplification, and all seven survive on an analytical tab in the workbook the analyst brings to the room, where they answer questions if questions come.

13.9 Communicating Uncertainty to Executives

The artifact now has a hierarchy, a sequence, a set of sentences, and a controlled amount of interaction. This section addresses the property those instruments most easily destroy. Chapter 10 produced a forecast with a prediction interval and insisted that the interval was the honest part; Chapter 11 produced a measured lift with a confidence interval and a break-even line beneath it; Chapter 9 produced a threshold derived from asymmetric costs. All three of those results have a range attached, and all three are about to be presented to a reader who wants a number. What happens in that collision is the subject here, and the guide's position is stated at the outset so that the argument can be read against it: the interval is what makes the recommendation defensible, not what weakens it, and an analyst who strips it has not made her advice more useful but has transferred the risk from her own credibility to her employer's balance sheet.

Start with why the stripping happens, because it is not usually cowardice. Hullman (2020) surveyed ninety visualization authors and interviewed thirteen influential designers about why uncertainty so rarely appears in published work, and documented a consistent tension: authors describe uncertainty as valuable and omit it anyway, because it is hard to represent well, because they fear an audience will read a range as incompetence, and because a distribution takes space and explanation that a point estimate does not. Those are real constraints and they are the design problem, not an excuse to skip it. Spiegelhalter et al. (2011) surveyed the available forms for communicating uncertainty about the future to non-technical audiences and reached a conclusion this section adopts: the choice of form matters more than the choice of statistic, and forms that let a reader see the range as a set of possible worlds outperform forms that ask her to interpret an abstraction.

Three practical moves follow, and the go/no-go dashboard uses all three.

The first is to express the interval in decision units rather than in statistical ones, and to say what the interval is before converting it. StyleCraft's FY2027 outlook is $1.25 million of chain contribution — $0.51 million already banked between July and November, plus a central estimate of $0.74 million for the seven months still to come — with the part still open running from $0.62 million to $0.86 million. That sentence is accurate and it means nothing to a chief executive in the three seconds it will get. Two conversions make it mean something, and the chapter's position is that a dashboard should carry both, because each answers a different question. The end-to-end width of the range still open is $240,000 of contribution, and $240,000 divided by the $137,000 of annual contribution a

mature store produces is 1.8 stores. Relative to the central estimate, the same range is approximately plus or minus $120,000, which is roughly plus or minus 0.9 mature stores. The end-to-end figure describes how far apart the two ends of the plausible range are; the plus-or-minus figure describes how far the outcome might sit from the number the plan is being built on, and quoting only the first overstates how uncertain the central estimate is. The annotation the labs build says both: *the range still open spans $240,000 end to end — about 1.8 mature stores' annual contribution, or roughly plus or minus 0.9 around the estimate.*

Notice what is absent from that arithmetic, because its absence was designed in. There is no margin in it. The forecast is stated in contribution and the denominator is stated in contribution, so the conversion is one division rather than a multiplication followed by a division. That is not a convenience; it is what keeps an unestablished claim out of the chain. Converting a chain-level *revenue* interval into store-equivalents would require multiplying it by a margin, and the only margin available is the 46 percent computed on the suburban and resort population — which is not the chain's margin, and assuming it is would bury an unsupported assumption inside an arithmetic step where no reader would ever find it. Selecting and fitting the model on contribution removes the step and the assumption together, which is the general principle: **when a conversion requires an assumption you cannot establish, change what you forecast rather than disclosing the assumption and proceeding.** Disclosure is the second-best answer and is available when the first is not. This is the *error in decision units* point of Table 8.5 applied to a range rather than to a point, and it is also where a careless analyst introduces a new error, which is almost always a units error, so the conversion is shown on the canvas in steps rather than asserted as a result, and the denominator is named as well as used.

The second is to draw the range as a ribbon rather than as a bar, and to build the ribbon from bounds that vary across the horizontal axis. A shaded interval region around a forecast line reads as a set of plausible futures and is decoded quickly; error bars on a bar chart are decoded with systematic biases that Correll and Gleicher (2014) measured, including a within-the-bar bias in which points inside a bar are judged more likely than points outside it at the same distance from the mean, and a tendency to read the interval's endpoints as a hard boundary rather than as a level of a continuous distribution. Their contribution was not only the diagnosis: they showed that gradient and violin encodings improved judgments on the same tasks, which is the reason this chapter prefers a graded region to a discrete marker. Nested regions — an 80 percent ribbon inside a 95 percent ribbon, each labeled — do not display the full probability density, and it is worth being exact about what they do and do not do: they communicate graded levels of uncertainty, showing that outcomes nearer the center are more concentrated than outcomes near the edges, without claiming to show how much more. The labs build the two-interval version because the contrast between the inner and outer region is what teaches the distinction, and because building it correctly forces the student to bring both sets of bounds into Tableau at the forecast's own grain rather than asking the software for a band.

That last point is the one that separates a correct uncertainty view from a plausible-looking one, and it deserves its own statement, because it is where the first draft of this dashboard failed. A reference band in Tableau is defined between two values at the table, pane, or cell scope, which produces a horizontal region across a pane rather than a region whose upper and lower edges move with the horizontal axis (Tableau, 2026a). A prediction interval that widens with the horizon is not a horizontal region. It has a different lower and upper value in every month, which means the bounds must exist as fields at month grain before Tableau is asked to draw anything, and the region must be constructed from those fields rather than requested from the Analytics pane. Tableau's own forecasting feature will draw a widening band, but it draws the band belonging to the forecast Tableau itself computed, at the single prediction interval currently configured (Tableau, 2026g); this chapter deliberately imports an externally produced forecast and displays two intervals at once, so neither convenience is available. The construction the labs use is a **polygon**: the region's outline is emitted upstream as an ordered list of vertices — along the lower bound left to right, back along the upper bound — and Tableau is asked only to connect them. Two nested intervals become three non-overlapping regions rather than two layered ones, so nothing can hide behind anything. Lab 13.1 Part D builds it, and the same construction draws the predecessor range in Part B, so the technique is learned once and applied twice. What it will not let a student do is the thing the tool makes tempting: drop two bound measures side by side on Rows and expect a filled region between them. Separate continuous measures on Rows occupy separate axes, stacking measures inside one pane requires Measure Values and Measure Names (Tableau, 2026h), and every segment

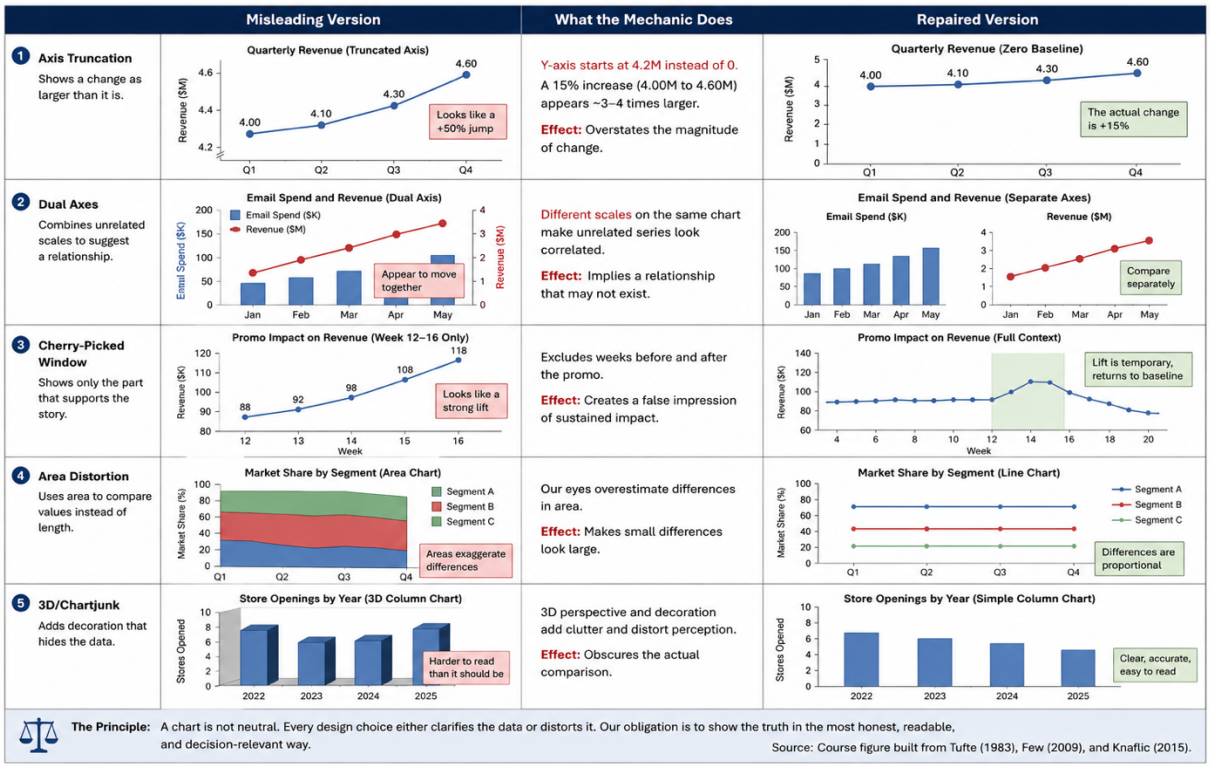
stacked that way shares one Marks card and therefore one opacity, because opacity is a property of the card rather than of a measure (Tableau, 2026i).

Figure 13.4

How an Interval Ribbon Is Actually Built

Figure 12.4 The Five Misleading-Chart Mechanics (with StyleCraft Examples and Repairs)

Each row shows (left) the misleading version, what the mechanic does to the reader, and (right) the repaired version.



The third practical move is to pair the range with the decision rule it implies, which is where Chapter 2's asymmetric-cost discipline finally earns its keep in a room. A range alone invites paralysis; a range with a stated threshold and an asymmetry invites a decision. The go/no-go version is the finance partner's own arithmetic returned to him, and getting it right requires care, because the two sides are easy to state in units that do not compare. The downside of *declining* a site that would have worked is not the $137,000 of annual contribution a mature store produces; that figure is contribution *before* the fixed footprint the site would also have carried, so the quantity actually forgone is the net — roughly $137,000 less $92,000, or about $45,000 a year, before build-out, taxes, and working capital. The downside of *signing* a site that never reaches coverage is likewise not the full $92,000; it is the part of the footprint the store's own contribution fails to cover, plus whatever is irreversible — the lease term, the fit-out, and the cost of exiting — and that last component is the one with no upper bound anywhere on this dashboard. Stated in comparable units the asymmetry runs toward caution rather than balance: the annual regret from a bad signing is bounded below by the unrecovered footprint and can exceed it by the exit cost, while the annual regret from a good site declined is closer to $45,000 and is partly recoverable, since a declined site can often be revisited and a signed one cannot be unsigned. That is the reasoning that makes *hold* the risk-appropriate default for a site whose comparable evidence is absent or unfavorable, and it is why the recommendation is conditional on the ramp evidence rather than a blanket judgment about the market. It is also why the paragraph on the screen states the comparison qualitatively and shows both subtractions: neither side is known precisely, and a screen that set $92,000 against $137,000 as though they were comparable would be doing arithmetic the case does not support. Stating that reasoning is what turns the uncertainty view from a hedge into the argument's load-bearing element.

CONCEPT

Hedging Without Paralysis

There is a form of honesty that is indistinguishable, in a meeting, from having nothing to say. It sounds like this: "The results are suggestive but the interval is wide and there are limitations, so it is hard to draw a firm conclusion." Everything in that sentence may be true and it is nonetheless a failure of the analyst's job, because the decision is being made on Friday whether or not the analyst helps, and an executive who receives no recommendation does not therefore make no decision — she makes it on the basis of whoever in the room is most confident, which is a worse process than the one the analysis was supposed to improve. The alternative is not false confidence.

It is a recommendation stated with its conditions attached, which has a reliable four-part shape: *here is what I recommend; here is the evidence it rests on; here is the range I cannot narrow, in units you care about; and here is what would have to be observed for me to change the recommendation.* The fourth part is the one that converts hedging into usefulness, because it turns the uncertainty into a monitoring plan rather than a disclaimer.

"We recommend signing the five sites matched to comparables that reached four-week-average coverage, holding the site matched to Scottsdale and the two with no qualifying comparable, and revisiting all three in eight weeks when the Wave 4 stores reach the age at which the mature stores' paths had already separated" is a conditional recommendation that a chief executive can act on Friday, and it is more honest than a confident number, not less.

13.10 From Dashboard to Decision

The artifact is built and its uncertainty is stated. Two things remain before December 2, and both are about closing the gap between what the analyst intends and what actually happens in a room. The first is the memo. The second is the test.

Take the memo first, because students consistently treat it as an afterthought and it is the deliverable most likely to survive the meeting. Charts and marks do not by themselves constitute a durable recommendation. A dashboard can certainly contain one — a callout, an annotation, a decision rule written on the canvas, a text object stating what the analyst advises — and the go/no-go dashboard's title band does exactly that. What a dashboard cannot do is carry the conditions, the scope, and the reversal criteria that make a recommendation actionable and auditable, because those are prose and there is no arrangement of marks that says *unless*. The pairing this guide asks for is therefore a one-page written recommendation accompanying every decision-supporting visual artifact, structured as the four-part shape of Section 13.9's concept box and carrying, in an appendix, the views that came off the canvas during decluttering. Three properties make it work. It is written before the meeting, which forces the analyst to discover whether she can actually state the recommendation — a recommendation that will not survive being written down will not survive being questioned. It is left behind, which means it is what circulates to the people who were not in the room and what is consulted in March when someone asks what was decided and why. And it states the reversal conditions, which is what makes the analyst accountable in a way a screen never is, since a screen that was wrong can always be reread as merely displaying.

Now the test, which is this chapter's verification theme and the reason the labs end where they do.

DEFINITION

Usability Test

A usability test of a visual artifact is a structured observation in which a representative reader, who has not seen the artifact, is given a task or a question and observed while attempting it, with the analyst recording what the reader looks at, what she says, what she asks, and what she concludes — and refraining from explaining, guiding, or defending. Its purpose is to measure the artifact's communication rather than the reader's competence: a point at which the reader hesitates, misreads, or asks a question the artifact should have answered is evidence of a possible mismatch among the artifact, the task, and the audience, and is investigated as a candidate defect in the artifact rather than dismissed as a property of the reader.

Because a first impression cannot be had twice, a reader who has seen an artifact can no longer supply a fresh first-impression measurement of it, though she remains available for later task-completion and revision testing.

Source: Adapted from Krug (2014) and Nielsen and Landauer (1993).

In other words, this is predict-then-verify with a human being as the instrument, and it is the only form of verification in this chapter that the analyst cannot perform alone. The protocol is short. Predict, in writing, what the reader will say the artifact is about, which question she will ask first, and where she will hesitate. Show the artifact for three seconds and record the takeaway. Show it again for ninety seconds with one task — "tell me what you would decide and why" — and record what she reads, in what order, and what she asks. Then stop, and grade the predictions. What makes the protocol hard is not the mechanics but the discipline of silence: the reflex to explain the chart the moment the reader misreads it destroys the measurement and is nearly irresistible, which is why the lab makes the analyst write her predictions down first, so that there is something at stake in shutting up.

How many readers are needed is a question with a well-known answer that is frequently overstated in both directions. Nielsen and Landauer (1993) modeled the discovery of usability problems as a function of the number of test participants and found sharply diminishing returns, which is the origin of the widely repeated "five users" heuristic. The model's parameters were estimated on particular systems and user groups, and it does not establish five as a universal number for every interface, audience, or task, nor does it license testing once and declaring victory. What this course asks for is smaller and more defensible: a formative usability check on one or two peers before a meeting, which is enormously better than none, with the second reader seeing the revised artifact rather than the same one. A check of that size finds the gross hierarchy and language failures and does not validate that the artifact will work for executives — it is a way of catching what the analyst cannot see, not evidence that the artifact is finished. The analyst who tests nobody is relying on the assumption that she is a representative reader of her own work, which she is definitionally not.

The classroom version of the test is a structured exchange rather than an observation, and it is the instrument Project #2 runs.

DEFINITION

Critique Protocol

A critique protocol is a structured procedure for obtaining and giving feedback on a visual artifact, in which the reviewer proceeds in a fixed order — first reporting what the artifact appears to say, then asking what decision it is meant to support, then identifying specific elements that impede the reading, and only then proposing changes — and in which the author listens without defending until the sequence is complete. The order is the mechanism: reporting the takeaway before learning the intent yields an uncontaminated measurement, and separating observation from prescription keeps the exchange focused on the artifact's effect rather than on the reviewer's taste.

Source: Adapted from Krug (2014) and Few (2006).

The protocol's value is that it makes the feedback usable. Unstructured critique of visual work reliably degenerates into preference — the reviewer would have used a different color, would have put the chart on the left, does not like the font — and preference is unfalsifiable and therefore unhelpful. A protocol that requires the reviewer to state the takeaway first produces a fact the author cannot argue with, and a protocol that requires her to name the element responsible before proposing a fix produces a change the author can evaluate. It also protects the reviewer, since "I read this as saying the new stores are failing" is an observation about herself rather than an accusation about the author's work, and observations of that kind are much easier to deliver to a peer.

One closing note connects the section back to the decision it serves. The purpose of all of this — the memo, the test, the protocol — is not to make the analyst's artifact better in the abstract. It is to reduce the probability that a correct analysis produces a wrong decision, which is the specific failure mode Part III exists to prevent and which is invisible in every measure of analytical quality that Chapters 4 through 11 supplied. A model with a good holdout error that leads to a bad lease decision has failed. That failure happens in the last two feet, it happens fast, and the only defense against it is to find out, before the meeting, what a person who is not the analyst actually sees.

13.11 AI as a Communication Assistant

Chapter 12 observed that a chart's mechanics and its meaning are produced by the same act, which is why an AI-drafted chart cannot be audited the way an AI-drafted model can. Communication work sharpens that observation into something more uncomfortable. An assistant asked to write a chart title, a dashboard headline, or a paragraph of recommendation prose produces language, and language is the medium in which a claim is made — which means that in this chapter the assistant is not drafting an instrument whose output the analyst will interpret. It is drafting the interpretation itself. The gap between what a view supports and what a sentence about that view asserts is the entire subject of Sections 13.6 and 13.15, and an assistant will close that gap fluently, confidently, and without any access to the evidence, because closing gaps fluently is what it does.

That is also, and not coincidentally, where the assistant is genuinely useful. Language is the part of this work most analysts find hardest and slowest, and an assistant that produces eight candidate headlines in fifteen seconds is offering something real. The division of labor this section installs is therefore narrower than in previous chapters rather than broader: the assistant may generate candidates, and the analyst must verify every one against the view it sits on, one sentence at a time, before any of them reaches a canvas.

The failure modes are specific, and the first is the chapter's signature.

The overclaiming title is the modal failure and the most consequential. Asked to title a view, an assistant will reliably produce a sentence that is more interesting than the view, because more interesting sentences are better writing and the assistant is optimizing for writing. The overclaim takes three recognizable shapes. It substitutes a causal or predictive verb for a descriptive one, so that "The four mature suburban and resort stores' four-week moving averages first reached the weekly fixed-cost threshold in weeks 43, 46, 51, and 57" becomes "the suburban format pays for itself within a year" — a claim about the future, about all such stores, made from four observations, using a word the view's measure does not support, and quietly dropping the store that took fifty-seven weeks. It generalizes the population, so that a statement about four mature stores becomes a statement about the format. And it imports a number the view does not contain, usually because the analyst pasted the chapter's findings into the prompt along with the view description and the assistant, quite reasonably, used them. The defense is mechanical and is the AI in Practice routine below: verify each sentence against the marks in the view directly beneath it, in the view's own units, with no other evidence admitted.

The confident recommendation is the second, and it is the same failure moved from the title bar into prose. Asked to draft a recommendation paragraph, an assistant will produce something decisive, because decisive prose is what the genre calls for and hedged prose reads badly. What it will not do, unasked, is carry the conditions — the interval, the threshold, the reversal criterion — and what it will frequently do is convert a conditional recommendation into an unconditional one during the rewrite, on the grounds of clarity. An analyst who hands over

"sign five, hold three, revisit in eight weeks" and receives back "the data supports proceeding with the expansion" has watched the assistant delete the finding and keep the tone.

The disappeared uncertainty is the third and follows from the second. Asked to make a paragraph "clearer" or "punchier," an assistant removes the interval before it removes anything else, because the interval is the clause that complicates the sentence. This is worth predicting explicitly before any such request, and it is worth checking by diffing the two versions rather than reading the new one, since the missing clause is invisible in a text that reads well.

The layout without a hierarchy is the fourth. Asked to propose a dashboard layout, an assistant produces a plausible arrangement — often a neat grid, sometimes with a row of headline numbers along the top — that reflects conventional dashboard appearance rather than any judgment about which view the decision turns on, since it has no access to that judgment unless the prompt supplies it. The output is the first draft of Section 13.1 with better spacing. The fix is to supply the KPI hierarchy of Section 13.4 in the prompt as an input and ask the assistant to implement it, at which point the task becomes mechanical and the assistant does it well.

The generic audience is the fifth. Prompts that do not specify the reader produce artifacts for a composite executive who does not exist — some finance vocabulary, some marketing vocabulary, a level of detail calibrated to nobody. Pasting the audience analysis of Section 13.2 into the prompt changes the output substantially and costs one paragraph.

The invented finding is the sixth and the most dangerous in this chapter specifically, because it arrives dressed as help. Asked to write dashboard copy from a set of view descriptions, an assistant will occasionally supply a connective sentence that asserts a relationship between two views — "the higher order value explains the faster coverage" — that no view establishes and that the analysis may in fact contradict. It reads as synthesis. It is confabulation, and it is caught only by the rule that every sentence must be traceable to a specific view.

The accessibility and provenance omissions are the seventh and eighth, and they are omissions rather than errors: nothing in a request for a headline or a layout will produce a note about the filter state, a statement of grain, alternative text for a screen reader, or a check on contrast, and none of these will be volunteered.

Where the assistant earns its place, the list is genuinely long. Generating a dozen candidate assertion titles for one view, so that the analyst chooses rather than composes, is the highest-value use in the chapter and inverts the usual difficulty of the task. Rewriting a paragraph for a stated reader — "rewrite this for a chief executive who has not read the analysis and will spend ninety seconds on it" — is a task assistants perform well and analysts perform slowly. Producing the calculated-field syntax for a dynamic title that tracks a parameter, or the Prep expression that reshapes four bound columns into the band-outline vertices Lab 13.1 Part D needs, is boilerplate — though the vertex order must still be checked by eye, because an assistant will produce a plausible ordering that does not close. Drafting alternative text for each view, which accessibility increasingly requires and almost nobody writes, is a natural fit. Converting a statistical statement into decision units — the store-equivalents arithmetic of Section 13.9 — is arithmetic the assistant will do correctly if given the inputs and will invent if not, so the inputs must be given. And the adversarial use remains the best one: handed a finished dashboard's title, subtitle, and view list, and asked what a reader could reasonably conclude that the underlying views do not support, an assistant produces a list that will contain at least one item the author did not see, in under a minute.

The governing instrument extends the audit lineage a final time. Table 8.5's three transferable points and Table 12.5's three visualization points both continue to apply — a communication artifact is built on views, and a view that fails V1 does not become acceptable because its title is good. Table 13.6 adds what communication makes necessary, and its three points are ordered by how often they are violated.

Table 13.6

The Communication Supplement to the Audit (run with Tables 8.5 and 12.5)

Supplemental point	The check	Fails when
C1. Every sentence is earned	For each title, subtitle, annotation, and line of recommendation prose, name the view that supports it and confirm the claim is verifiable from that view's marks in that view's units; confirm the verb is one the evidence hierarchy of Section 11.10 licenses; confirm every number in a sentence appears in a view or in the certified metrics file the view is built from	A title asserts a cause or a payback over a descriptive view; a headline generalizes four observations to a format; a number in the copy appears nowhere in the marks; a connective sentence asserts a relationship no view establishes
C2. The reader's takeaway matches the intended one	State the intended takeaway in writing before testing; run the three-second test on a representative reader; compare	The reader names a different subject; the reader reports the artifact is "about the new stores doing badly"; the reader cannot state any takeaway
C3. What was removed is disclosed	List every simplification made for the audience — views cut, categories grouped into "other," windows shortened, filters defaulted, filter scope narrowed, qualifications moved into tooltips — and confirm each is either immaterial to the decision or stated on the canvas	The default filter state is restricted and unlabeled; a qualification that changes the recommendation lives only in a tooltip; a category that would change the ranking is inside "other"

AI IN PRACTICE

Draft the Titles, Then Prove Them

The chapter's verification theme as a five-step routine, run on every artifact whose language an assistant touched. Step one is the request, written to separate generation from judgment: "Here is a description of one view from a dashboard: it plots store-age weeks on the horizontal axis and the four-week moving average of weekly contribution on the vertical, one line per store, for the four suburban and resort stores open more than eighteen months, drawn on a complete store-week scaffold so that weeks with no orders appear as zero rather than being skipped, with a horizontal reference line at the $1,769 weekly contribution that carries the $92,000 annual fixed footprint. The first week in which each line reaches the reference is week 43, week 46, week 51, and week 57 respectively. Write twelve candidate titles as complete sentences.

Do not use any number that I have not given you. Do not assert a cause, a forecast, a payback period, or a claim about stores other than these four. For each candidate, state in a second line exactly which marks in the view a reader would have to look at to verify it." That last clause is the whole routine in miniature: it forces the assistant to produce its own audit trail, and a candidate whose verification line is vague is a candidate that cannot be verified. Step two is the prediction, made before reading the candidates: write down how many of the twelve you expect to overclaim, and in which of the three shapes — causal or predictive verb, generalized population, imported number.

> Step three is the audit: take each candidate, cover the description you supplied, look only at the built view, and decide whether a reader could confirm the sentence from the marks; reject on the first failure rather than repairing, because repairing a candidate reintroduces the analyst's own knowledge into a sentence that must stand without it. Step four is the grade: compare the overclaim count and shapes against your prediction and record both misses and false alarms, since a systematic error in your model of the assistant is more useful than any single caught title. Step five is the reader, and it is the step that does not exist in Chapter 12: show the surviving title on its view to someone who has not seen either, ask what it says, and confirm that her paraphrase is the claim you intended rather than the stronger one next door.
>
> File the exchange per Appendix D. And carry the rule that gives the routine its force: an assistant can write a sentence, but only the analyst can sign it, and the sentence at the top of a chart in an executive meeting is signed whether or not anyone thought about it that way.

13.12 Hands-On Application in Tableau

The preceding sections specified a reader, chose an artifact, built a hierarchy, laid it out, gave it sentences, emphasized and subtracted, bounded its interactivity, and stated its uncertainty. This section builds it. Lab 13.1 constructs the go/no-go dashboard in four parts: the decision brief, the certified inputs, and the KPI specification written before anything is opened; the building of three views, two of them rebuilt rather than reused from Chapter 12 and one with no ancestor at all; the assembly of the canvas with its containers, its one action, and its declared fixed state; and the uncertainty view that carries the fiscal-year outlook into the room. Lab 13.2 tests it: a three-second test and a ninety-second task on a peer, followed by a structured critique exchange, a revision, and the one-page recommendation memo.

The convention of Chapter 12 continues. A Tableau lab's equivalent of a code cell is a build path, so the boxed element is labeled **Build** and is followed by the same plain-language account of input, transformation, and output. AI assistants may draft any build path or any sentence in this section, and every output — including every sentence — is predicted before it is produced.

13.12.1 The Chapter's Inputs and Where They Come From

This chapter uses more declared quantities than any other in Part III, and it uses them in assertion titles, which is the one place a number must never arrive without a derivation. Before any build path, therefore, the inputs are named, sorted into the two kinds they belong to, scoped to the population they describe, and reconciled.

Start with the population, because everything else inherits it and because getting it wrong is the error that survives every other correction. The decision on December 2 is eight sites under letter of intent, and all eight are **suburban and resort** formats. StyleCraft's fifth wave also contains flagship sites, and they are deliberately not in this meeting: they are not yet at letter of intent, they carry a different cost structure, and finance has supplied no footprint figure for them. They are a separate decision on a separate benchmark, and saying so in one sentence in the source band is what keeps this dashboard's evidence from being read as evidence about them. Every *store-level* figure on the screen therefore describes the same population — the footprint, the margin, the predecessor stores, the Wave 4 cohort, and the prospective sites are all suburban and resort — and the two chain-level figures that remain, the fiscal-year outlook and the chain averages the supporting views draw as references, are labeled as chain wherever they appear. What is true without qualification is that **nothing on this screen describes the flagship format**. The single most useful check a student can run on her own artifact is to write the population beside each number and confirm that every entry reads either "suburban and resort" or "chain," and that no entry reads both.

One definition has to come before both kinds, because every figure on the screen sits downstream of it. **Weekly contribution** in this chapter is `line_revenue` less `line_cost`, summed within a store and a store-age week, and it carries weight only if `line_cost` is what the name implies. In the certified extract it is: `line_cost` carries all variable product and fulfillment cost assigned to the order line — merchandise cost, inbound freight,

payment processing, and pick-and-ship — so weekly contribution is what remains after the costs that vary with the sale and before the fixed footprint the threshold represents. That is the property the whole dashboard depends on, because the comparison it draws is between contribution and rent, staffing, and allocated overhead, and a quantity that had not yet absorbed variable fulfillment cost would be compared against a footprint it cannot honestly be compared against.

Check this rather than inheriting it, and say what you find. If your extract's `line_cost` carries merchandise cost alone, the quantity you have computed is **gross profit**, not contribution: relabel it throughout, and either subtract the remaining variable costs before the comparison or state plainly that the threshold is being compared against a figure that has not yet absorbed them. The distinction changes nothing visually and changes what every title on the screen is entitled to say, which makes it exactly the kind of definition that gets inherited silently and should not be. The same applies to the 46 percent: it is a contribution margin only because the numerator is contribution, and it would be a gross margin if the numerator were gross profit.

The first kind of input is a **finance planning assumption**: a figure supplied by a person, for a purpose, that the extract cannot produce and does not contain. There is exactly one, and it is the $92,000 annual fixed footprint — rent, staffing, and allocated overhead for a new StyleCraft store in the suburban and resort formats, which finance prices together. It is a forward-looking planning number carrying the finance partner's judgment about lease and staffing costs in the markets under letter of intent, and it is used here as declared. Two consequences follow and both belong on the canvas. It must be attributed to finance rather than to the data, and it must be declared before the results are inspected, because a threshold chosen after the crossing weeks are known is not a benchmark but a description — which is the reference-line discipline of Section 12.9 applied to the one number the whole dashboard turns on.

The second kind is a **certified derived quantity**: a figure computed from `orders_certified` by a stated calculation, published in the companion repository, and reproducible by any student who runs the calculation. Four files carry them.

- `chapter13_store_week_scaffold.hyper` — one row per physical store per store-age week, carrying weekly revenue, weekly contribution, the four-week moving average of contribution, and the predecessor range's lower and upper bound at that week; with the band-outline rows of Part A step 3 unioned in. This is the extension to contribution of the scaffold Lab 12.1 Part C built.
- `chapter13_dashboard_metrics.csv` — the scalar figures of Table 13.7, together with each store's first observed and first four-week-average coverage week.
- `chapter13_fy27_outlook.csv` — the fiscal-year contribution outlook at month grain, its interval bounds, and its certified horizon totals.
- `chapter13_prospective_sites.csv` — the eight sites under letter of intent, each with its proposed format, its trade-area population-density band, its median-household-income band, its matched existing comparable, and the flags recording eligibility and match.

The outlook file needs a paragraph of its own, because it is not a file Chapter 10 produced and because two decisions were made in building it that a reader is entitled to see.

The first decision is the **measure**. Chapter 10 forecast chain revenue. This chapter forecasts chain **contribution**, by rerunning Chapter 10's time-ordered model-comparison procedure on the daily contribution series and using the best-performing contribution model. Note the wording: not *the same model refit*, but *the same selection procedure rerun*. The specification that won on revenue may not win on contribution, whose variance structure, seasonal amplitude, and response to promotional weeks all differ, and assuming the winner transfers is the kind of inheritance this chapter spends a section warning against. The reason is arithmetic honesty rather than preference. The dashboard's uncertainty annotation converts an interval into store-equivalents, and the denominator of that conversion is a mature store's annual *contribution*. Converting a chain-level *revenue* interval into contribution would require multiplying it by a margin, and the only margin available is the 46 percent computed on the suburban and resort population — which is not the chain's margin, and assuming it is would be an unestablished claim buried inside an arithmetic step where no reader would find it. Forecasting contribution directly removes the step and the assumption together. The 46 percent margin still appears in this chapter, but only where it belongs: converting the

weekly contribution threshold into the weekly *revenue* a suburban or resort store must take to reach it, which is a within-population conversion.

The second decision is the **horizon**, and it is where the audit of the previous draft was hardest to answer honestly. The certified transaction extract ends June 30, 2026, so the model's forecast origin is July 1, 2026 and its horizon runs to June 30, 2027. The meeting is December 2. Disclosing that five months of the horizon have already elapsed is better than concealing it, and it is not good enough: July through November are no longer uncertain future values, and presenting them inside a prediction interval two days before eight leases price would price risk that has already resolved.

The file therefore carries an **actual-to-date plus forecast-to-go** structure, and it is worth following the seam carefully, because the two halves come from different places. July through November are **actuals from finance's monthly close** — a separate certified source, at month grain, which the company has long since reported and which no analyst's transaction extract is needed to produce. December through June are the selected contribution model's forecast, from the July 1 origin, with the interval re-derived on the December-through-June sum from the same simulated paths. The fiscal-year figure is therefore the sum of a known quantity and an uncertain one, and only the second half has a range around it.

Two limitations follow and both belong on the canvas. The model has not seen July through November: it was fit on daily data ending June 30, and the analyst cannot refit it on months for which no transaction-level extract exists yet. The range on the remaining seven months is therefore conditional on a fit that stops in June, which is why *refit when the extract refreshes* is one of the memo's reversal conditions rather than a housekeeping note. And the five elapsed months supply something free that the previous version threw away: the model predicted them, finance reported them, and comparing the two is a calibration check available at no cost. If the reported actuals sit near the middle of what the model expected for those months, the interval on December through June deserves more weight; if they sit outside it, the interval deserves less, and the memo should say which. The general lesson is that a forecast presented long after its origin is not merely stale — it overstates its own uncertainty about the part that has already happened, and it wastes the evidence that part contains.

One note on the status of everything below, because it governs how the table is used. Every figure in Table 13.7 is a **certified case value**: computed by a stated calculation from the certified extract, published in the companion repository with the answer key, and quoted in this chapter's titles, memo, and summary as the case's established result. None of them is provisional, and none is quoted with a hedge, because a chapter that hedged its own numbers could not teach assertion titles at all. What the student owes is reproduction, not adjustment. Recompute every row before building anything; where your figure differs from the published one, the difference is the finding and it goes in the audit — you do not move your number toward the table, and you do not move the table toward your number without saying so.

Two rows deserve a word about sensitivity, which is a different thing from provisionality. The coverage weeks and the Wave 4 classification both depend on the smoothing rule declared below, and a store sitting close to the range's edge at week 17 can be classified differently under a four-week average than under raw weekly contribution. That is a property of the measure rather than a weakness in the figure, and it is why the rule is declared before it is applied and why the reproduction step exists.

Table 13.7 reconciles every figure the dashboard displays and names the population each one describes.

Table 13.7

Reconciliation of the Figures Used on the Go/No-Go Dashboard

Figure	Population	Source and calculation	Verified result
Annual fixed footprint	New suburban and resort stores	Finance planning assumption, declared before results were inspected; not derived from any extract. No figure supplied for the flagship format	$92,000 per store per year
Weekly fixed footprint	Same	$92,000 ÷ 52	$1,769.23

Figure	Population	Source and calculation	Verified result
Weekly contribution	Suburban and resort stores	`orders_certified`; Σ(`line_revenue` − `line_cost`) within store and store-age week, where `line_cost` carries all variable product and fulfillment cost	Revenue after variable cost, before the fixed footprint
Contribution margin	Suburban and resort stores	`orders_certified`; Σ(`line_revenue` − `line_cost`) ÷ Σ(`line_revenue`), on the same cost definition	46 percent
Weekly revenue at coverage	Same	$1,769.23 ÷ 0.46	$3,846.15
First observed coverage week	The four mature suburban and resort stores	Scaffold; first store-age week whose weekly contribution reaches $1,769.23	Weeks 38, 42, 44, and 51
First four-week-average coverage week	Same	Scaffold; first store-age week whose four-week moving average of weekly contribution reaches $1,769.23	Weeks 43, 46, 51, and 57
Mature-store annual contribution	Same	`orders_certified`; trailing 52-week contribution for each of the four stores over the year ending June 30, 2026, then averaged across the four. Not a weekly mean annualized, which would inherit whatever seasonality the sampled weeks happened to carry	Approximately $137,000
Predecessor range	Same	Scaffold; minimum and maximum of the four-week average across the four stores at each store-age week 3–25, the average being null before week 3	23 bound pairs, one per week; see file
Wave 4 position at week 17	The eight Wave 4 suburban and resort stores	Scaffold; count above, within, and below the range at store-age week 17, on the four-week average	2 above, 5 within, 1 below; the classification most sensitive to the smoothing rule
FY2027 contribution outlook	Chain	`chapter13_fy27_outlook`; July–November reported by finance, plus December–June forecast from the contribution model selected under Chapter 10's validation design, fit on certified daily history through June 30, 2026	$1.25 million ($0.51m reported + $0.74m central estimate)
80 percent range, forecast-to-go	Chain, December–June only	Simulated paths summed over the seven remaining months, then 10th and 90th percentiles taken. Not reconstructible by summing the monthly bounds	$0.62 million – $0.86 million
FY2027 outlook range	Chain	Actual-to-date added to each end of the forecast-to-go range	$1.13 million – $1.37 million

Figure	Population	Source and calculation	Verified result
Interval in store-equivalents	—	($0.86m − $0.62m) ÷ $137,000 end to end; half that around the estimate	1.8 stores end to end; ±0.9 around it
Average order value	Chain; NYC Suburban	`orders_certified` (Chapter 12); SUM(`line_revenue`) ÷ COUNTD(`order_id`)	$108 chain; $156 suburban
Repeat-order share	Chain; suburban	`orders_certified` (Chapter 12); orders after the customer's first purchase date ÷ distinct orders	47 percent chain; 24 percent suburban
Prospective sites	The eight sites under letter of intent	`chapter13_prospective_sites`; eligibility and matching rules of Part A step 6	All 8 eligible; 6 matched (5 to comparables that reached four-week-average coverage, 1 to Scottsdale); 2 unmatched

The labs use `orders_certified` and `lines_certified`, the two extracts produced by the Tableau Prep flow of Lab 12.1 Part A, which reconcile to $4,882,247.60 in total revenue across 45,206 orders and 96,844 order lines; confirm those totals against the answer key in the companion repository before treating any of them as a target. Students who did not build the extracts should do so before beginning; nothing in this chapter re-derives them.

One definition remains, and it must be fixed before it is measured, because the phrase students reach for first is the wrong one and the rule they reach for second is too weak to carry a lease. The measure this dashboard is built on is not a payback period. Payback ordinarily means the time taken to recover an accumulated investment, and nothing on this screen accumulates anything: the view compares a store's weekly contribution against a week's share of its fixed cost.

DEFINITION

Weekly Coverage: Observed and Four-Week-Average

First observed weekly coverage is the earliest store-age week in which a store's weekly contribution reaches the declared weekly share of its annual fixed footprint. **First four-week-average coverage** is the earliest store-age week in which the four-week moving average of that contribution reaches the same threshold. Both are coverage milestones rather than payback periods: they say when a store carried its own weekly fixed cost, and say nothing about whether it recovered its build-out.

The distinction is not pedantic, and which one is made primary is a decision about what the artifact is for. A single promotional week, a holiday week, or one large corporate order can lift a store above the threshold once without establishing anything durable, and a lease is a multi-year commitment that should not be justified by a one-week milestone. This chapter therefore makes **four-week-average coverage the decision measure** and keeps observed coverage on the same view as a supporting milestone, drawn as a lighter marker, because the gap between the two is itself informative: a store whose two weeks are close crossed on strength its neighboring weeks shared, and a store where they are far apart crossed once on something the surrounding weeks did not repeat.

Be exact about what the four-week measure does and does not establish, because the language students reach for overstates it. It is **not** a claim that the store held coverage, stayed above the line, or established durable coverage. A trailing average can be lifted over a threshold by one strong week and drop below it the following week, and the milestone says only that at some point a four-week window averaged at or above the line. It is a smoothed first crossing, and it is preferred to the raw one because it is harder to produce by accident — not because it certifies persistence. A rule that did certify persistence would have to be declared as such: the fourth week of the first run of four consecutive raw weeks at or above the threshold, or the first week of a declared run of consecutive averaged weeks.

Either is defensible and neither is used here; what is not defensible is using the smoothed rule and describing it in the language of the stricter one.

Whichever rule is used, it must be declared before it is applied, and it must be computed rather than read off a plotted line. A line crosses a threshold somewhere between two weekly marks; the eye will pick the wrong one, and a value a student obtains by inspection is a value the next student cannot reproduce.

Source: Course concept developed for this guide, informed by the reference-line discipline of Section 12.9 and the baseline discipline of Section 2.7.

Note what the store-week scaffold does and does not do for these two measures, because the scaffold's value is easy to overstate in exactly one direction. Store age is derived from the opening date rather than from the row count, so a week absent from the transaction extract does not renumber the weeks after it: dropping a zero-order week before week 38 does not make week 38 arrive earlier, and **first observed coverage returns the same answer with or without the scaffold**. What the scaffold genuinely changes is threefold. It changes the *drawn path*, since Tableau otherwise connects across the gap and renders a fortnight of nothing as a straight line through it. It changes the *predecessor range*, because a minimum computed across four stores at each week is biased upward wherever one of those stores' zero weeks has been dropped, which narrows the region at its floor and can move a Wave 4 store from inside it to below it, or the reverse. And it is a **precondition** for four-week-average coverage, because a four-week moving average computed over an extract with holes in it averages the wrong four weeks. Making four-week-average coverage the primary measure is therefore also what gives the scaffold its work in the primary view rather than only in the comparison.

13.12.2 Lab 13.1, Part A: The Decision Brief, the Scaffold, and the KPI Specification

Almost nothing is drawn in this part. That is the point of it, and it is the part students skip and then rebuild the dashboard twice.

BUILD

The certified inputs and the written specification

Step 1. Extend the Chapter 12 scaffold to contribution. Open the Prep flow from Lab 12.1 Part A. On the branch that produced the store-week scaffold of Part C, add `Weekly Contribution` as the sum of `line_revenue` less the sum of `line_cost` within each store and store-age week, and set it to zero — not null — for any store-week in which the store was open and observable but recorded no orders. Leave it null where the store was not yet open or the window had not closed.

Add `Contribution 4wk Avg` as a Prep multiple-row calculation, specified completely rather than described: **group by** the store identifier, **order by** `Store Age Weeks` ascending, **compute using** `Weekly Contribution`, over a window of **the current row and the previous three**, and return null until four observable scaffold weeks exist for that store rather than averaging a short window (Tableau, 2026k). Every one of those five properties has to be stated, because a moving calculation with any of them left to a default computes a different number and the difference is invisible in the output. Extend the scaffold's store-age range from 0–25 to 0–64, since four-week-average coverage arrives later than observed coverage and the view must reach it. Output as `chapter13_store_week_scaffold`.

Step 2. On the scaffold, compute the predecessor range at each store-age week: the minimum and maximum of `Contribution 4wk Avg` across the four mature suburban and resort stores at that week, output as `Band Low` and `Band High`. Use the smoothed measure, not the raw weekly one, and use it because the primary view uses it: the decision rule is four-week-average coverage, and a comparison view that plotted raw weekly paths against a raw weekly range would evaluate the new stores on a measure the recommendation does not turn on. Identical axes do not make differently smoothed quantities comparable. These are per-week values, and note where they begin: the four-week average is null until a store has four observable scaffold weeks, so the range exists for store-age weeks **3 through 25** and not before.

That is 23 pairs, not 26, and the first three weeks of the axis carry no region at all — which is correct and should be left visible rather than patched, because a comparison band over a store's first fortnight would be built from a window that does not exist. Compute them in Prep, not in the view.

Step 3. Reshape those bounds into band-outline rows, which is what Tableau will actually draw. For each band, emit one row per vertex in drawing order: walking the lower edge left to right, then the upper edge right to left, so that the sequence traces the outline of the region and closes on itself. Four columns carry it — `series_type` set to `band`, `band_level` naming which band the vertex belongs to, `path_order` numbering the vertices from 1, and `band_value` holding the vertex's vertical position at that `store_age_week`. For the predecessor range over weeks 3–25 that is 23 vertices at `Band Low` from week 3 to week 25, followed by 23 vertices at `Band High` from week 25 back to week 3, or 46 rows.

Vertex 1 and vertex 46 share a horizontal position and not a vertical one — one sits on the lower edge and one on the upper — and Tableau closes the shape by drawing the segment between them, which is the left edge of the region. If a literally coincident final vertex is preferred, emit a 47th row repeating vertex 1. Union those rows into `chapter13_store_week_scaffold` beside the store rows, which carry `series_type` set to `store`. One extract, two kinds of row, one shared horizontal axis.

Step 4. On the scaffold, compute `First Observed Coverage Week` per store as the minimum store-age week whose `Weekly Contribution` reaches 1769.23, and `First 4wk Coverage Week` as the minimum store-age week whose `Contribution 4wk Avg` reaches the same value. Output both to `chapter13_dashboard_metrics` **and join them back onto every scaffold row for their store**, on the store identifier. Both are per-store scalars, and View 1 is built on the scaffold rather than on the metrics file, so a field that lives only in the metrics file cannot be dropped onto a shelf without a relationship the lab never establishes. Joining in Prep is the simpler of the two available answers; relating the metrics file to the scaffold on the store identifier is the other, and either must be chosen deliberately.

Reproduce every row of Table 13.7 and record any mismatch as a finding.

Step 5. Revalidate the outlook model on its new target before using it. Chapter 10 selected its model on the *revenue* series, and the model that wins on revenue is not necessarily the model that wins on contribution — a series with a different variance structure, different seasonality amplitude, and a different response to promotional weeks. Rerun Chapter 10's evaluation ladder on the daily contribution series without shortcuts: refit the naive and seasonal-naive baselines; refit each candidate under the same time-ordered rolling-origin design and the same holdout; compare errors in contribution dollars rather than in percentages; and either confirm that Chapter 10's specification still wins or adopt the contribution-series winner and say so. Publish the comparison table beside `chapter13_fy27_outlook`.

Until that ladder has been rerun, the outlook is a well-explained figure produced by a model validated for a different target, and no title should quote it.

Step 6. Build `chapter13_prospective_sites` from the eight letters of intent: one row per site, carrying its proposed format, its trade-area population-density band, its median-household-income band, its matched existing comparable, and two flags. Write both rules down before applying either. **Eligibility:** a site is eligible for comparison against this dashboard's evidence only if its proposed format is suburban or resort; a flagship site requires a separate footprint, a separate margin, and a separate predecessor range, none of which exist here. **Match:** among eligible sites, a site matches an existing store when all three of the following hold — the same format, the same trade-area population-density band, and the same median-household-income band.

Tie-break: where more than one existing store qualifies, take the one with the smallest standardized distance across density and income, and record every candidate alongside the selection, because "closest comparable" is not reproducible unless the distance and the tie-break are both written down. The conjunction is deliberate. Format alone would match every eligible site to something, since every eligible site is suburban or resort and so is every existing comparable, and a rule that matches everything classifies nothing. Record the resulting counts. The memo's recommendation is a rule before it is a count, and a count without a rule is an opinion with a number attached.

Step 7. Write the decision in one sentence, naming the decision-maker, the choice, and the deadline. The form is: *StyleCraft's chief executive must decide by Friday whether to execute eight suburban and resort leases in full, in part, or not at all.* If the sentence requires an "and," the artifact is being asked to support two decisions and one of them belongs elsewhere.

Step 8. Write the audience analysis of Section 13.2 as five lines: who reads it, what decision, how long, what she knows, what would change her mind. Keep it to five lines; the constraint is the exercise.

Step 9. Write the intended takeaway as one sentence — the sentence you want the chief executive to say after three seconds. Seal it. Lab 13.2 grades it, and a takeaway written after the test is not a prediction.

Step 10. Complete Table 13.8 for the artifact: for every view, its tier, the reader's question it answers, the measure and grain, and the reference it carries — pairing each row with the assertion title drafted for it per Table 13.3. Seven rows is the target and eight is the maximum.

Step 11. For each of the five views inherited from Chapter 12, write one sentence naming the view it derives from and stating what must change in the rebuild; "nothing must change" is not an available answer, because if it were, Section 12.2 would be wrong. For the two purpose-built views, write one sentence instead stating the question they answer that no Chapter 12 view addressed, and why the pre-read did not need it.

Step 12. List every view and every control from the pre-read that will *not* appear, with one line apiece on where it goes instead — the memo's appendix, a tooltip, the workbook you bring to the room, or nowhere.

Input is the certified order-grain extract and the Chapter 12 Prep flow; the transformation is the extension of the store-week scaffold to contribution and its moving average, its reshaping into band-outline rows, and the computation of the derived quantities upstream of any view; the output is one extract carrying store rows and band rows, a metrics file, a site file, and a written specification. Steps 1 through 3 are the ones students are most tempted to skip, and skipping them changes the answer rather than the effort. `orders_certified` contains no row for a week in which a store took no orders, so a view built directly on it draws a line straight across the missing week, and a four-week average computed on it averages whichever four rows happen to be adjacent rather than the four weeks the calendar contains.

Step 6 deserves one further note, because the two rules it asks for do different jobs and students routinely collapse them into one. Eligibility asks whether this dashboard's evidence *applies* to a site at all, and it is answered by format, because the footprint, the margin, and the predecessor range are all suburban and resort quantities. Matching asks, among sites the evidence applies to, whether a specific existing store is a fair comparable, and it is answered by format *and* trade-area characteristics together. A site can be eligible and unmatched — the evidence applies to it in principle, but no existing store resembles it closely enough to say what its ramp will look like — and two of the eight are exactly that. Two failure modes sit on either side of the rule and both are worth naming, because students hit one or the other. A disjunctive match rule matches everything and therefore discriminates nothing, which is where the counts in Table 13.7 would collapse to eight and zero. A rule that folded eligibility into matching would let a trade-area resemblance stand in for a format the evidence does not cover, matching a flagship site to a suburban comparable while a later step declared that forbidden. Keeping the two rules separate and the second conjunctive is what makes the five-and-three split reproducible.

Table 13.8

The Go/No-Go Dashboard Specification

View	Tier	Reader's question	Measure and grain	Reference carried	Derives from
1. Coverage path	Primary	Do these stores cover their own cost, and when?	Four-week moving average of weekly contribution; one line per mature suburban or resort store, one mark per scaffold store-week	Horizontal line at $1,769.23; first four-week-average coverage week labeled on each store, first observed coverage week marked more faintly	New; the pre-read did not answer the finance partner's question
2. Wave 4 ramps	Primary	Are the new ones on track?	Four-week average of weekly contribution by store age; one line per Wave 4 store, eight in all	The predecessor range drawn as a polygon whose outline is emitted in Prep; shared vertical scale with View 1	Lab 12.1 Part C, store-age facet, collapsed from 17 panels to one frame

View	Tier	Reader's question	Measure and grain	Reference carried	Derives from
3. Order value	Supporting	What is different about these customers?	Average order value at order grain, by `home_metro`; one mark per metro	Chain average at $108.00, entered as a constant from the metrics file	The Section 12.10 grain demonstration, drawn as a view for the first time. Lab 12.1 View A is *not* the ancestor: it answered the distribution question at order grain against the chain median, and Table 12.1 named a bar of the mean as the wrong form for that purpose
4. Category mix	Supporting	Is it a mix story or a pricing story?	Category revenue share by metro; one mark per category within metro	None; shares are self-referencing	Lab 12.1 View E
5. FY outlook	Supporting	How much of this is a guess?	Monthly chain contribution, July 2026 – June 2027, actual through November and forecast thereafter, with 80 and 95 percent bounds at month grain on the forecast months only	A labeled horizon-total block carrying the outlook and the range still open; interval width annotated in store-equivalents, both framings	`chapter13_fy27_outlook`, re-encoded
6. Repeat share	Diagnostic	What are you not telling me?	Repeat-order share by `store_type`; one mark per store type	Chain share at 47 percent, entered as a constant	Lab 12.1 View B
7. The outlier	Diagnostic	Which sites would you not sign?	Four-week average of weekly contribution by store age for the one Wave 4 store below the range	The same band polygon as View 2	Lab 12.1 Part C, one panel extracted

VERIFICATION CHECK

Before you run: predict what the dashboard will fail at, and write down three predictions sealed with the takeaway from step 9. Predict which of the seven views will turn out to be unnecessary — one of them will, and identifying it after the build rather than before is the ordinary outcome, so the prediction is worth grading. Predict the reading order a fresh reader will actually follow, as a numbered list of the seven views, and note that your prediction and your intended order are two different objects; if they match perfectly you have almost certainly predicted your intention rather than the reader. Predict the first question the chief executive will ask after the three seconds, and check whether the artifact answers it anywhere.

After you run: reconcile every row of Table 13.7 against your own outputs, and write the population beside each figure — every one of them should read "suburban and resort," "chain," or "derived," and any figure whose population you cannot name is a figure that has not been checked. Confirm that $92,000 divided by 52 is $1,769.23 and that this divided by 0.46 is $3,846.15. Confirm that the contribution margin you compute on the suburban and resort store population rounds to 46 percent, and report the unrounded figure. Confirm that the scaffold contains exactly one row per physical store per store-age week in range, that zero-order weeks carry zero rather than null where the store was open, and that no store carries a non-null contribution for a week before it opened.

Confirm the band rows are 46 for the predecessor range, that `path_order` runs 1 to 46 without gaps, that the first and last vertices sit at store-age week 3 — at different heights, one on each edge, with Tableau's closing segment joining them, since an outline whose ends do not meet at a common horizontal position will not fill cleanly — and that no band row exists for weeks 0 through 2, which is the visible consequence of the four-week window's null rule.

Then run the three comparisons that show what the scaffold is actually for. First, recompute `First Observed Coverage Week` on `orders_certified` without the scaffold and confirm it returns the *same* four weeks, since store age comes from the opening date and dropping a week does not renumber the ones after it — a check designed to fail to find a difference, and worth running for that reason. Second, recompute `First 4wk Coverage Week` both ways and report whether it moved, and by how many weeks, for each store; the four-week window cannot be computed correctly on an extract with holes in it. Third, recompute `Band Low` at every store-age week without the scaffold and report the largest upward shift in the region's floor.

Finally, report the gap between each store's observed and four-week-average coverage week and say in one sentence what a large gap would mean about that store's early trading.

Investigate if: any store's observed coverage week is week 0 or week 1, which usually means the threshold was applied to revenue rather than to contribution; any store's four-week-average coverage week is *earlier* than its observed week, which is arithmetically impossible and means the moving average was computed on the wrong window; the predecessor range is flat across store age, which means `Band Low` and `Band High` were computed at table scope rather than per week; the site file records a match for an ineligible site, which means the two rules of step 6 have been collapsed into one; the contribution margin differs from 46 percent by more than a point, in which case the weekly revenue threshold changes and Table 13.7 must be rebuilt before any title quotes it; or `line_cost` turns out to carry merchandise cost alone, in which case every use of the word *contribution* in this chapter is wrong and must be corrected before anything is drawn.

13.12.3 Lab 13.1, Part B: Rebuild, Do Not Reuse

Chapter 12's Section 12.2 asserted that an analytical view should never drift into a communication artifact without being rebuilt. This part is where the assertion is cashed, and the instruction is literal: open new worksheets. Duplicating the pre-read's sheets and editing them preserves every default the analytical version was entitled to and the explanatory version is not, and the ninety seconds saved is reliably spent three times over.

BUILD

View 1 — the coverage path (primary)

1. New worksheet on `chapter13_store_week_scaffold`, not on `orders_certified`. The scaffold is the source for every view that plots a weekly path, for the reason Part A gave.

2. Drag `Store Age Weeks` to **Columns** (continuous). Drag `Contribution 4wk Avg` to **Rows** with aggregation Min — the scaffold is already at store-week grain, so the aggregate returns the single scaffold value. Set the Marks card to Line, and `store_name` to **Detail**.

3. Filter to the four mature suburban and resort stores. Filter `Store Age Weeks` to 0–64 so the frame reaches the latest four-week-average coverage week, and record that choice in the caption.

4. Analytics pane → **Constant Line** on the vertical axis at 1769.23. Edit the label to read "Weekly contribution carrying the $92,000 annual footprint (finance assumption, suburban and resort)" rather than the default value. A constant is correct here and a computed reference would be wrong: the threshold is external and was declared before the results were inspected, per Section 12.9 and Tableau (2026a).

5. Color: place `store_name` on **Color** and assign one accent hue to all four, not four different hues. The reader's comparison is between the lines and the reference, not among the lines.

6. Remove the legend and add direct labels: **Label** → Line Ends → Label end of line, showing `store_name`.

7. Use `First 4wk Coverage Week` — joined onto every scaffold row in Part A step 4, which is why it is available here without a relationship — to place the annotations, rather than clicking the plotted line. Add it to **Detail**, then annotate the mark at each store's four-week-average coverage week: right-click that mark → **Annotate** → **Point** → label with the week number only ("week 43"). Four annotations, no more. Confirm that each annotated week is the first week at or above the reference on the moving-average line and not merely a week near it.

8. Add the supporting milestone without letting it compete, and note that it needs a Marks card of its own — a field on **Detail** cannot change one point on a line into a different shape. Create `Observed Marker` as `IF [Store Age Weeks] = [First Observed Coverage Week] THEN 1769.23 END`, which returns a value at exactly one week per store and null everywhere else. Note the constant rather than the smoothed value: placing the marker on the *threshold line* at the right week says that this is where a single week first crossed, while placing it on the moving-average path at that week would invite the reader to take it for another measurement on the smoothed series, which it is not. Place it on **Rows** beside the line measure, right-click its axis → **Dual Axis**, → **Synchronize Axis**, and hide the duplicate header.

Set that second Marks card to **Shape**, choose an open circle, reduce its size, and give it a lighter tint of the accent hue; separate measures receive separate Marks cards, which is what lets a line and a point shape coexist with different formatting (Tableau, 2026h). Then right-click the shape axis → **Move Marks to Front** so the markers are not lost under the lines (Tableau, 2026j). Leave them unlabeled, and edit the tooltip to say what the mark is — "first single week at or above the threshold" — so a reader who does hover is not left guessing. The gap between the hollow marker on the threshold and the annotated week on the curve should be readable without any of that.

9. Format: remove the worksheet border, remove the column gridlines, remove the axis title on the horizontal axis (the assertion title will say it), keep the vertical axis title with its units.

10. Title: replace the default with the assertion title from Table 13.3, and add a subtitle in smaller gray type stating the grain and both rules: "One line per store; four-week moving average of weekly contribution, on a complete store-week scaffold. Annotated week = the first week whose four-week average reached the threshold; hollow marker on the threshold line = the earlier single week that first crossed it."

Input is the store-week scaffold filtered to four stores; the transformation is none beyond the filter, because the moving average and both coverage weeks were computed upstream; the output is a four-line chart with one reference, four annotations, four supporting markers, and a sentence. This is one of the two views with no Chapter 12 ancestor — the pre-read never asked the finance partner's question, so there is nothing to rebuild — and it is therefore the cleanest available demonstration of what an explanatory view is when nothing is inherited. Seven of the ten steps are communication decisions rather than construction: one hue instead of four, direct labels instead of a legend, annotations instead of none, a restricted and justified window, a stripped frame, a second milestone deliberately subordinated rather than omitted, and a title that states a finding. The remaining three are not communication work at all; they are the difference between a number a reader can reproduce and a number the analyst read off a line.

BUILD

View 2 — the Wave 4 ramps against the predecessor range (primary)

1. New worksheet on `chapter13_store_week_scaffold`. `Store Age Weeks` on **Columns** (continuous). Filter `Store Age Weeks` to 0–25 — the full range the store lines occupy, and one week further would leave the region trailing off at an edge with nothing to close against. The band's vertices cover only weeks 3 through 25, for the reason Part A step 2 gave, so the first three weeks of the frame carry lines with no region behind them. Leave that gap; it is where the four-week average does not yet exist.

2. Build the region before the lines, because the region is the harder object and the lines are trivial. Its edges move week by week, which is what a Tableau reference band cannot produce: a band is defined between two values at table, pane, or cell scope and renders as a region of constant height across the pane (Tableau, 2026a). Nor can the two bound fields simply be dropped side by side on **Rows** and stacked — separate continuous measures on Rows produce separate axes, and stacking measures within one pane requires **Measure Values** and **Measure Names** (Tableau, 2026h), which then leaves every stacked segment sharing a single Marks card and therefore a single opacity, since opacity is a mark property of the card rather than of a measure (Tableau, 2026i).

The construction that avoids both problems is a **polygon**, drawn from the outline the Prep flow already emitted.

3. Create two calculated fields that split the unioned extract into its two kinds of row: `Band Value` as `IF [series_type] = "band" THEN [band_value] END`, and `Store Value` as `IF [series_type] = "store" THEN [Contribution 4wk Avg] END`. Note the measure: the four-week average, matching View 1 and matching the range's own construction. Nulls are dropped by each mark type, so the same extract feeds both layers without a blend.

4. Restrict the line layer to the intended cohort — and do it with a calculation rather than with an ordinary filter, because an ordinary filter on `Store Wave` or `store_type` would also discard the band-outline rows, whose store fields are null. Create `Cohort Filter`:

```
Cohort Filter =
  [series_type] = "band"
  OR ( [series_type] = "store" AND [Store Wave] = "Wave 4" )
```

Place it on **Filters** and set it to True. The subtitle claims eight Wave 4 stores; this is the step that makes the claim true, and without it the frame contains every store in the chain while the sentence above it says otherwise.

5. Place `Band Value` on **Rows** as `MIN([Band Value])`. Set the Marks card type to **Polygon**. Drag `Path Order` to the **Path** shelf as a continuous dimension, and `Band Level` to **Detail**. The vertices now connect in the order Prep numbered them, tracing the lower edge left to right and the upper edge back again, and the outline closes on itself and fills.

6. Set the polygon's color to a pale tint of the neutral gray. One mark, one fill, no base to hide — which is the reason this construction is preferred to a stacked area, and the reason it extends without modification to the nested case in Part D.

7. Now add the lines. Place `Store Value` on **Rows** to the right of `Band Value`, right-click its axis → **Dual Axis**, and set that Marks card to Line with `store_name` on **Detail**. Right-click either axis → **Synchronize Axis**, then hide the duplicate header. Confirm the synchronization by reading one store's value against both axes before hiding.

8. Fix the layer order explicitly rather than relying on which axis is which. Right-click the line axis and choose **Move Marks to Front** (Tableau, 2026j), then confirm visually that every line is drawn over the region. Dual-axis layers can obscure one another, and this control exists precisely because the default is not something to depend on.

9. Set the vertical axis range to match View 1 exactly. This is the step that makes the adjacency of Section 13.5 work, and Tableau will not do it for you.

10. Color: one accent hue for all eight Wave 4 lines. Do not assign eight hues to eight stores.

11. Direct-label only Scottsdale, the one store tracking below the range. The other seven are the seven-of-eight the title refers to and are left unlabeled.

12. Title: the assertion title from Table 13.3. Subtitle: "Four-week average of weekly contribution. Shaded region = the range the four mature suburban and resort stores occupied at the same store age, week by week. Eight Wave 4 stores, all suburban or resort, 17 weeks old."

Input is the unioned extract of store rows and band-outline rows; the transformation is none inside Tableau beyond splitting the two row types and restricting the line layer; the output is a single frame carrying the comparison that Chapter 12 needed seventeen panels to make. Three things about this construction generalize and are worth naming. The first is the governing idea: **a region with a moving edge is a shape, not an aggregation**, so it is described upstream as an ordered set of vertices and Tableau is asked only to draw it. That is the same discipline as the floating-bar reference of Table 12.2 — a mark measured from an explicitly supplied reference rather than from zero — and it is why the chapter keeps computation in Prep. The second is that polygons compose: regions that do not overlap are simply separate sets of vertices distinguished by `Band Level`, which is exactly what the three edge-to-edge regions of Part D need. The third is step 4's lesson, which arrives whenever two kinds of row share one extract: **a filter written against a field that only one kind of row possesses will silently delete the other kind**, so the filter has to be written as a calculation that names both. A stacked **Measure Values** area with an opaque white base reaches a similar result and is the documented alternative if the polygon path proves awkward in your build; it costs the gridlines behind the region and cannot carry per-segment transparency.

Note also the trade being made, and make it consciously: the small multiples of Lab 12.1 Part C were the better analytical instrument, because they showed every store's individual path; this single frame is the better communicative instrument, because it shows one comparison and a ten-minute reader can only make one. Both are true, the second is not an improvement on the first, and an analyst who cannot say which object she is building will build the wrong one.

BUILD

View 3 — order value with the chain reference (supporting)

1. New worksheet on `orders_certified`. `home_metro` on **Rows**, and a calculated field `Average Order Value` as `SUM([line_revenue]) / COUNTD([order_id])` on **Columns**, formatted as currency with no decimals. Note what is being built and what is not. The pre-read's View A plotted the *distribution* of order values by metro, against the chain median, because the question then was whether suburban orders are uniformly larger or larger and more dispersed — and Table 12.1 was explicit that a bar of the mean is the wrong form for that question. The question here is different: the executive needs the level, once, against a chain benchmark. Same calculation as Section 12.10's grain demonstration, first time it has been drawn, and a form Chapter 12 would have rejected for Chapter 12's question.

2. Sort descending. Confirm the axis includes zero — bars encode by length, and Section 12.6's obligation does not relax because the audience is an executive.

3. Analytics → Constant Line at 108 → label "Chain average $108." Enter it as a constant taken from the certified metrics file rather than as an average computed from the four displayed marks, for the reason Lab 12.1 View B gave: the chain figure is computed on the pooled denominator and is not the unweighted mean of the metros.

4. Color: the suburban bar in the accent hue, all others gray. One emphasis, per Section 13.7.

5. Direct-label the bar values; remove the value axis, which now duplicates the labels.

6. Title: the assertion title from Table 13.3. Subtitle: "One bar per metro; average order value computed at order grain; full certified window."

Input is the order-grain extract; the transformation is a ratio of two aggregations by metro; the output is a four-bar chart carrying one emphasized comparison against one drawn reference. Step 6's subtitle is not a formality: this is the number Section 12.10 showed can be computed three ways, the dashboard is asserting the order-grain version over the full certified window, and a reader who later reconciles it against a colleague's $50.41 needs the grain on the canvas to resolve the discrepancy in the analyst's favor.

VERIFICATION CHECK

Before you run: predict what will have to be removed from each view's Chapter 12 ancestor, as a list, and grade the list afterward against what you actually removed.

After you run: run C1 from Table 13.6 on each of the three titles — cover your own knowledge, look only at the marks, and confirm that a reader could verify the sentence. For View 1, confirm the annotated weeks sit on the moving-average line rather than on a raw weekly value; confirm each hollow marker sits on the threshold line rather than on the curve, at its store's first single-week crossing; and confirm the markers are visibly subordinate to the annotations. For View 2, count the lines in the frame and confirm the count is eight and not seventeen; then confirm that "seven of the eight" and "within or above" are both things a reader could establish from the marks rather than things she must take on trust.

Verify the region is not flat: select three widely separated store-age weeks — say weeks 4, 12, and 24 — and confirm from the tooltip that the region's upper and lower vertices differ at each and are not the same pair repeated. Confirm the region begins at week 3 and not at week 0, and that the gap at the left of the frame is the three weeks in which no four-week average exists rather than a rendering fault; confirm too that it closes cleanly at both ends rather than trailing off. Finally, place Views 1 and 2 side by side at the size they will occupy on the dashboard and confirm that their vertical axes are identical.

Investigate if: the region renders as a rectangle of constant height, which means a reference band was used instead of the polygon; the region renders as a scribble or a bow tie, which means `Path Order` is on Detail rather than on the **Path** shelf, or is being treated as a discrete dimension; the region begins at week 0, which means the four-week average was computed over a short opening window instead of returning null; the region disappears when the cohort filter is applied, which means the filter was written against a store field rather than as the calculation of step 4; the lines are drawn under the region, which means **Move Marks to Front** has not been applied to the line axis; or the two axes are not synchronized, in which case every visual comparison in the frame is meaningless and no reader will be able to tell.

13.12.4 Lab 13.1, Part C: Assembly, Emphasis, and a Declared Fixed State

The views exist. This part makes them one artifact, and it is where the hierarchy of Section 13.4 either becomes visible or stays in the analyst's head. It is also where this chapter's most counterintuitive design decision gets made, which is that the finished executive artifact carries no filter control at all.

BUILD

The canvas

1. Dashboard → New Dashboard. Set **Size** to Fixed, at the projection resolution of the room the meeting is in. Fixed is correct here because the consumption condition is known and the hierarchy is expressed in relative area; Automatic, Range, and device-specific layouts are the correct choices for artifacts published to unknown displays, and the choice should be recorded in the specification rather than defaulted (Tableau, 2026e).

2. Drag a **Vertical** container onto the empty canvas. Inside it, place four **Horizontal** containers: a title band, a primary band, a supporting band, and a diagnostic band. Build everything inside containers; floating objects will not hold their relationships when anything changes.

3. Title band: a **Text** object carrying the decision, not the data — "Wave 5: eight suburban and resort leases, decision Friday" — and, in smaller type, the date and the analyst's name. The person who signs the artifact should be on it.

4. Primary band: View 1 at roughly 40 percent of the canvas width and View 2 beside it at roughly 35 percent, axes aligned. This is the primary tier and it occupies the upper-left path.

5. Supporting band: Views 3, 4, and 5 as a half-height row beneath the primaries, each about a third of the width.

6. Diagnostic band: Views 6 and 7 at roughly half the height of the supporting row, with a lighter title treatment so their prominence matches their tier.

7. Add padding: select each container and set outer padding to at least 8 pixels, inner to 4. The whitespace is the grouping mechanism of Section 13.5 and the default of zero destroys it.

8. Add the source band: a Text object at the foot carrying the extract names, the certified window, the forecast origin, the populations, the state the screen is fixed in, and the refresh date. One band for the screen, not one per view. It should read something like: "Sources: `orders_certified`, `chapter13_store_week_scaffold`, `chapter13_fy27_outlook`, `chapter13_prospective_sites`. Physical stores only; online and app orders excluded. Coverage evidence covers suburban and resort formats only; flagship sites are a separate decision on a separate benchmark. Outlook: July–November reported by finance; December–June forecast from the contribution model selected under Chapter 10's validation design, fit on certified daily history through June 30, 2026. No filters applied; this screen shows the complete certified population."

9. Add no filter controls. This is a design decision and it needs to be made deliberately rather than by omission, so write the reasoning into the specification. The first draft carried seven. Five were removed because a ten-minute reader making a one-time decision was never going to use them and their visual presence made a false claim about what the screen was for. The remaining two were removed for a stronger reason, which is that no scope existed at which either was safe. A date-window control would move the marks on Views 3, 4, and 6 while leaving "$156," "$108," "24 percent," and "47 percent" standing in their titles and drawn in their reference lines, all four of which are full-window constants.

A `store_type` control is worse: applied to Views 1, 2, and 7 it can empty View 1 entirely, change the number of lines in a frame whose title says "seven of the eight," and — because the band-outline rows carry no store type — delete the predecessor region while leaving the sentence that describes it. Scoping a filter to selected worksheets narrows which sheets it touches; it does not keep a title true and it does not protect rows the filter has no value for (Tableau, 2026c). The honest number of filter controls on this artifact is zero, the state is instead declared in words in the source band, and all seven controls survive on a separate analytical tab in the workbook the analyst brings to the room.

10. Actions. **Dashboard → Actions → Add Action → Highlight**, source View **7**, target View **2**, running on hover, on `store_name`. The direction matters and is the point of the exercise. View 7 contains exactly one store, Scottsdale, and View 2 contains all eight, so every value the source can emit has a match in the target and the action fires for every possible selection. Reversed — source View 2, target View 7 — it would fire for one of the eight Wave 4 lines and do nothing, silently, for the other seven, because a highlight action operates on matching field values and a target that does not contain the selected value simply does not respond (Tableau, 2026b). View 1 is excluded from the action in either direction, since it contains only the four mature stores and shares no store name with the Wave 4 cohort at all.

Test the action by hovering Scottsdale's line in View 7 and confirming that its line separates from the other seven in View 2.

11. Parameter as navigation, with a title that tracks it. Create a string parameter `Metro Shown` listing the four metros and "All," defaulting to "All." On View 4, add a calculated field `Metro Filter` reading `[Metro Shown] = "All" OR [home_metro] = [Metro Shown]`, place it on Filters, and set it to True. Then create `Mix Title` as `IF [Metro Shown] = "All" THEN "Occasionwear is roughly three times the share of suburban revenue that it is of urban" ELSE "Occasionwear's share of revenue, " + [Metro Shown] + " only" END`, drag it to **Detail** — a calculated field cannot be inserted into a worksheet title unless it is present in the view — and then **Worksheet → Show Title → Insert →** `Mix Title` (Tableau, 2026f). Show the parameter control beside the view rather than anywhere that reads as a global setting.

12. Tooltips: on Views 1, 2, and 5, edit the tooltip to carry the store name or month, the store age, the weekly contribution or the month's bounds, and the grain sentence. Nothing load-bearing goes here, per Table 13.5.

13. Alt text: for each view, open the worksheet's **Accessibility** settings and edit the **alt text** so that a screen-reader user receives one sentence describing what the view shows and what its marks represent (Tableau, 2026d). Tableau's default alt text names the fields; replace it with the view's finding and grain.

Input is the seven worksheets; the transformation is spatial rather than numerical; the output is a single fixed-size screen whose geometry states the hierarchy and whose state is declared in prose rather than exposed in a control. Step 9 is the most consequential and the least obvious, and it is worth being clear about what it is and is not. It is not an argument that dashboards should not have filters; Table 13.5's first row still describes the conditions under which a filter communicates, and the analytical tab in the same workbook carries all seven. It is the application of that row's own test to this specific artifact: a filter communicates when it narrows a population the reader already understands and cannot falsify anything on the screen, and on a canvas where six of seven titles quote fixed constants, no control on offer met the second condition. Step 11 is worth pausing on for the mirror-image reason, because it is the one interactive element that survived. In its default state View 4 carries an assertion title, which is a two-metro comparison; in any single-metro state that comparison is no longer on the screen, so the calculation returns a label title instead. That is legitimate precisely because the assertion has already landed in the default state and the parameter is answering a follow-up rather than delivering the finding. A parameterized view whose *default* state carries a label title has hidden its finding behind a control, which is the failure the last row of Table 13.5 describes.

VERIFICATION CHECK

Before you run: write down the state the screen is fixed in — population, window, forecast origin, exclusions — and predict which assertion titles would become false if a `store_type` control were added and set to Resort only. Name the four figures a date-window control would have falsified.

After you run: run three checks on the assembled canvas before anyone else sees it. The squint test — step back from the screen or blur the image until the text is unreadable, and note which object still dominates; if it is not View 1, the hierarchy is in your head rather than on the canvas, and the fix is size and position rather than a bigger title. The declared-state check from C3 of Table 13.6 — reload the dashboard fresh and confirm in writing that the source band states the population, the window, the forecast origin, the exclusions, and the fact that no filter is applied, because a screen with no controls on it makes no claim about its own completeness unless the words are there.

And the interaction inventory — list every interaction you built, and for each one name the reader's question it answers, state what the reader who never touches it will miss, and confirm by testing that it actually fires. Any interaction whose answer to the second question includes a finding is a design failure, and the finding moves onto the canvas.

Investigate if: the highlight action appears to do nothing, which means the source and target views do not share the field values; the parameter's title reads as an assertion in a single-metro state; the source band describes a state the screen is not actually in; or the alt text still names fields rather than describing the finding.

13.12.5 Lab 13.1, Part D: The Uncertainty View

This part builds View 5 separately because it is the view most likely to be built badly, and because it is where Chapter 10's work either survives the trip to the meeting or quietly does not. It reuses the polygon construction of Part B with three regions instead of one, and it adds the structure that makes a forecast presented five months after its planning year began still worth presenting.

BUILD

View 5 — the fiscal-year outlook with two nested interval regions

1. Understand the file's structure before drawing anything, because it is what the whole view is about. `chapter13_fy27_outlook` covers July 2026 through June 2027 and carries two kinds of month. July through November are **actual**, taken from finance's monthly close: contribution is known and there are no bounds. December through June are **forecast**, from a model origin of July 1, 2026, and carry `point`, `lo80`, `hi80`, `lo95`, and `hi95`, with the horizon rows holding the December-through-June totals quantiled from the simulated paths. The fiscal-year total is therefore a known quantity plus an uncertain one, and only the second half has a range around it. The file also carries, for each actual month, what the model predicted for it — which is what the calibration note in Section 13.12.1 asks the student to check.

2. Reshape the forecast months into band-outline rows exactly as Part A step 3 did, and union them with the monthly series into one extract. Three regions are emitted, and they are deliberately made **non-overlapping** so that no drawing-order problem can arise: `Lower 95` runs between `lo95` and `lo80`, `Central 80` between `lo80` and `hi80`, and `Upper 95` between `hi80` and `hi95`. Each carries `series_type` = `band`, its `band_level`, a `path_order` tracing its lower edge left to right and its upper edge back again, and `band_value` at each `Order Month`. The monthly rows carry `series_type` = `actual` or `forecast`, with `Order Month` and `value`. Every bound exists at month grain, which is the property the whole view depends on, and no band row exists before December, which is the property that makes the picture honest.

3. New worksheet. `Order Month` on **Columns** as a continuous date, spanning July 2026 through June 2027 and no further. Do not extend the axis past the fiscal year; a long empty tail shrinks the regions' apparent width, which is the axis-scaling question Exercise 13.17.7 returns to.

4. Do not generate a new forecast inside Tableau. This dashboard communicates an already verified result; replacing it with a second, unexplained model would break the analysis-to-communication chain, and Tableau's forecasting feature in any case displays only the single prediction interval currently configured, which cannot produce the two nested intervals this view requires (Tableau, 2026g).

5. Define the two measures the view needs, so that each mark type receives only the rows it should draw:

```
Band Value   = IF [series_type] = "band" THEN [band_value] END

Series Value = IF [series_type] = "actual"
                  OR [series_type] = "forecast"
               THEN [value] END
```

6. Draw the regions. `MIN([Band Value])` on **Rows**, Marks card set to **Polygon**, `Path Order` on the **Path** shelf as a continuous dimension, `Band Level` on **Detail** and on **Color**. Assign the palest tint of the forecast hue to `Lower 95` and `Upper 95` and a visibly darker tint of the same hue to `Central 80`. Because the three regions are edge-to-edge rather than layered, the result reads as two nested intervals and nothing is hidden behind anything — which is the reason step 2 emitted three regions rather than two.

7. Add the series. Place `MIN([Series Value])` — the field defined in step 5 — on **Rows** as a second pill, right-click → **Dual Axis**, **Synchronize Axis**, hide the duplicate header, and set that Marks card to Line with `series_type` on **Detail** and on **Color**. Draw the actual months in the full forecast hue and the forecast months in a lighter tint of it, so that the change in certainty is visible in the line as well as in the regions. Note the consequence of putting `series_type` on Detail: it splits the series into two paths, so the line will break between November and December unless November is emitted a second time as an anchor row in the forecast segment.

Emit it, and emit the right value: the anchor repeats November's **reported actual** contribution, not the model's prediction for November, which the model produced from a July origin and which nobody is claiming as the month's outcome. Say why in the caption — the seam is communicated by the tint change and the seam line, not by a gap, and a gap invites the reader to think a month is missing. Do not attempt to draw one segment solid and the other dashed: a line's dash pattern is a mark property of the Marks card, set through **Path**, and it is not assignable to individual members of a dimension on that card (Tableau, 2026i).

8. Mark the seam. **Analytics → Constant Line** on the horizontal axis at December 1, 2026, labeled "Reported through November · forecast from here." This is the single most important non-data element on the view, because it is what tells the reader that the left half is history and the right half is a range. Note that this line marks where *reporting* ends, not where the model's origin sits: the model was fit through June 30 and has not seen the five actual months, and the subtitle has to say so.

9. Fix the layer order explicitly: right-click the line axis and choose **Move Marks to Front** (Tableau, 2026j), then confirm visually that the line is drawn over the regions.

10. Add the horizon total, which is the step most easily skipped and the one the assertion title depends on. The view plots months; the title states a fiscal-year figure, and no reader can verify a sum by looking at a line. Place a small text block inside the view's frame, in the upper left where it will be read before the marks, reading: "FY2027 chain contribution · $0.51m reported (Jul–Nov) + $0.74m forecast (Dec–Jun) = $1.25m · FY range $1.13m – $1.37m, of which $0.62m – $0.86m is still open." Take every figure from the horizon rows of `chapter13_fy27_outlook`, which were produced by summing each simulated path over the seven remaining months and then taking percentiles of the totals.

Do **not** compute them in the view by summing the monthly bounds; adding up the edges of monthly intervals overstates the width, which is the aggregation rule Chapter 10 established and which this view is the natural place to violate.

11. Annotate once, at the right edge of the central region, with both store-equivalent framings: "The range still open on the remaining seven months spans about $240,000 of contribution end to end — roughly 1.8 mature stores' annual contribution, or about ±0.9 stores around the estimate."

12. Title: the assertion title from Table 13.3. Subtitle: "Monthly chain contribution. Left of the line, reported actuals from finance's monthly close. Right of it, the central estimate with the central 80% and 95% of simulated outcomes, from the contribution model selected under Chapter 10's validation design and fit on certified daily history through June 30, 2026."

13. Do not add a second measure axis, a growth-rate line, or a target line. This view has one job.

Input is the fiscal-year outlook file unioned with its band outlines; the transformation is none inside Tableau, because the three regions were described upstream as ordered vertices; the output is a continuous monthly series, unshaded over the reported months, with an uncertainty region that begins in December and widens across the forecast horizon. The series itself does not widen; the region around it does, and saying it the other way is the kind of shorthand that becomes a reader's misreading. Steps 8, 10, and 11 are the view's entire communicative payload, and it is worth noticing that all three are sentences or single marks rather than encodings: the seam line says where reporting ends, the horizon block makes the title verifiable, and the annotation makes the width meaningful. None of the three does another's job, and a view carrying only the regions has drawn an honest picture above a title no reader can check.

Step 1 also settles a question the previous version of this lab left open, and it is worth stating because it generalizes past this case. A forecast is not a fixed object with a fixed horizon; it is a statement made from an origin, and the distance between that origin and the room is part of the evidence. When months of a planning horizon have already happened, they are actuals and belong on the left of a line, not inside a band. Presenting them inside the band does not merely look stale — it overstates the analyst's own uncertainty, prices risk that has already resolved, and produces a range wider than the decision actually faces.

VERIFICATION CHECK

Before you run: predict what a reader will say the shaded region means, and write the prediction down; the two failure modes to watch for are "the contribution will be somewhere in there" stated as a certainty, and "they don't really know" stated as a dismissal, and the seam line and the annotation are what separate them.

After you run: confirm no shaded region appears to the left of the seam line, which is the check that the actual months carry no bounds. Confirm the regions widen: read the tooltip at December and at June and confirm the central region is thicker at June; a region of constant thickness means the bounds were not brought in at month grain. Confirm the three regions meet edge to edge with no gap and no overlap at every forecast month — `Lower 95` should end exactly where `Central 80` begins — which is a property of the file and a check on the reshape. Then confirm the horizon block's figures against the file: $0.51 million actual plus a $0.74 million central estimate is $1.25 million, and the outlook range of $1.13 million to $1.37 million is the actual added to each end of $0.62 million to $0.86 million.

Then verify the store-equivalents arithmetic in the open rather than in your head. The width of the range still open is $860,000 minus $620,000, or $240,000 of contribution; divided by the $137,000 of annual contribution a mature store produces, that is 1.8 stores; halved, because the annotation quotes both, it is about ±0.9 stores around the estimate. Notice what is *not* in that calculation: no margin. The forecast is already in contribution and the denominator is already in contribution, which is the whole reason the model was selected and fit on the contribution series rather than inherited from the revenue one.

Now compute two wrong answers deliberately — multiplying the $240,000 by the 46 percent margin gives $110,400, or 0.8 stores, and dividing the $240,000 by the $92,000 footprint gives 2.6 — and state in one sentence what each of them is actually measuring and why neither belongs on a canvas.

Investigate if: a shaded region appears in July through November, which means the actual months were given bounds; the region begins at the first month of the fiscal year rather than at the seam line; the regions render as a scribble, which means `Path Order` is not on the **Path** shelf; the line is drawn under the regions, which means **Move Marks to Front** has not been applied; the horizon block's figures do not match the file; or the summed edges of the monthly intervals were used as the range still open, which Chapter 10's aggregation rule forbids and which overstates the width.

Finally, remove the three regions and the seam line, look at the view, and write one sentence describing what the dashboard would have claimed without them. Keep that sentence; it is the exhibit for Section 13.15.

13.12.6 Lab 13.2, Part A: The Three-Second Test and the Ninety-Second Task

The artifact is finished. Nothing so far has produced any evidence that it communicates, and this part supplies the first of it. Work with a partner from the course who is building a different artifact; you will test hers afterward.

BUILD

The test protocol

1. Retrieve the sealed takeaway from Lab 13.1 Part A, step 9, and the sealed predictions from its Verification Check. Do not reread the dashboard first.

2. Write three further predictions: the first thing your reader's eye will land on, the first question she will ask, and the one element you expect her to misread.

3. Seat the reader in front of a blank screen. Display the dashboard for three seconds. Remove it. Ask one question: "What was that about?" Write her answer verbatim. Do not paraphrase it, and do not respond to it.

4. Display it again with one task: "You are the chief executive. Tell me what you would decide and why. Think aloud." Give her ninety seconds. Record, in this order, what she looks at first, second, and third; every question she asks; every hesitation; and her stated decision.

5. Say nothing for the entire ninety seconds. Do not explain any view. Do not answer any question — write it down instead and say you will come back to it. This is the hardest instruction in the chapter and the one that determines whether the test produces data.

6. Only after the ninety seconds are up, answer her questions and discuss.

Input is a finished artifact and an uncontaminated reader; the transformation is ninety-three seconds of structured observation; the output is a verbatim record and a set of graded predictions. The record is the deliverable, not the discussion afterward: what the reader said in the first three seconds is a measurement of the artifact's hierarchy, and it is worth more than her considered opinion at minute five, when she has become an expert on a dashboard she saw

once. Note also what this check is and is not. It is a formative usability check on one reader, which finds gross hierarchy and language failures; it is not evidence that the artifact will work for a chief executive, and reporting it as though it were is the kind of overclaim Section 13.6 spent a page on.

VERIFICATION CHECK

Before you run: seal all six predictions in writing. A prediction written after the reader has spoken is not a prediction, and the entire value of the protocol is the gap between the two.

After you run: grade all six predictions explicitly, and write one paragraph on the pattern rather than on the individual misses. Three specific comparisons carry most of the information. Compare the reader's three-second takeaway to your sealed intended takeaway word by word: a mismatch in the *subject* of her sentence — she says the artifact is about new stores when you intended it to be about cost coverage — is a hierarchy failure and is repaired with size and position, while a mismatch in the *verb* — she says the stores are failing when you intended "on track" — is a title or emphasis failure and is repaired with language and color.

Compare her actual reading order to the order you predicted in Part A, and note every view she skipped entirely; a skipped view is not a view that failed to persuade, it is a view that was never read, and the two have different fixes. And compare the questions she asked against the questions Table 13.8 claims each view answers: a question the specification says is answered, asked anyway, means the view answers it in a way she could not find.

Investigate if: your reader restates your intended takeaway word for word, which is more often a sign that she has seen the artifact before or has been coached than a sign of a perfect design; she asks no questions at all, which usually means the task was not understood rather than that the artifact was; or every defect she surfaces is one you had already predicted, which means the prediction was written loosely enough to cover anything. Then state, in one sentence per defect, whether you will fix the artifact or accept the defect, and why — accepting a defect is a legitimate outcome, and recording it is what makes it legitimate.

13.12.7 Lab 13.2, Part B: The Critique Exchange, the Revision, and the Memo

The test produced defects. This part fixes them, and it adds the deliverable that carries the recommendation.

BUILD

The critique exchange

1. Swap artifacts with your partner. Each of you now critiques the other's, following the protocol of Section 13.10 in order and without skipping ahead.

2. Stage one, report: state what the artifact appears to say, in one sentence, before being told anything about it. Write it down before speaking.

3. Stage two, ask: ask what decision the artifact is meant to support and who reads it. Only now does the author speak, and only to answer these two questions.

4. Stage three, identify: name up to five specific elements that impede the reading, each as an observation about your own experience of the artifact rather than as a judgment — "I could not tell which of these two charts to read first," not "the layout is confusing."

5. Stage four, propose: for each identified element, propose one change. The author writes them down and does not defend, argue, or explain during any of the four stages.

6. Reverse roles and repeat.

7. Revise your dashboard against the combined evidence from Part A and the critique. For each change, record what evidence prompted it. For each piece of feedback you decline to act on, record why in one sentence.

BUILD

The recommendation memo

1. One page. Four parts, in this order: the recommendation, the evidence, the uncertainty, and the reversal conditions.

2. The recommendation is one sentence stating what should be done and to what scope, and it states the **rule** before any count. The conditional form of Section 13.9 is expected. It must also distinguish the two reasons a site is held, because they are different findings with different remedies: a site matched to a comparable that did *not* reach coverage has unfavorable evidence, while a site with no qualifying comparable has no evidence at all.

An imperative is not required — "We recommend signing the sites matched to comparables that reached coverage, and holding both the site matched to an unsuccessful comparable and the two with no qualifying comparable" is a recommendation, and so is the imperative form of the same sentence — but a sentence that states a finding rather than an action is not, and "the evidence supports the expansion" is a finding wearing a recommendation's clothes. A count may follow the rule; it may not replace it.

3. The evidence is one short paragraph per primary view, each stating the finding in decision units and naming the view it comes from.

4. The uncertainty is one paragraph containing the range still open in store-equivalents in both framings, the arithmetic that produced the conversion with its denominator named, the forecast origin and the share of the planning year already banked, the scope limitation — that every store-level figure on the screen describes the suburban and resort formats, that the outlook and the chain averages are labeled as chain, and that no footprint, margin, or predecessor range exists for the flagship sites in the pipeline — and the two facts on the diagnostic band: the 24 percent suburban repeat-order share against the chain's 47 percent, and the store below the predecessor range.

5. The reversal conditions are two or three observable events that would change the recommendation, each with a date or a threshold attached. At least one must name the refit that follows the next extract refresh, since the outlook's range rests on a model that has not seen the months already reported.

6. The appendix lists the views that came off the dashboard, with one line apiece on why, the finance assumption with its source named, the site-matching table from `chapter13_prospective_sites` showing how each of the eight sites was classified and by what rule, and the AI-Use documentation per Appendix D. Any count in the recommendation must be reconcilable to that table; a count the appendix cannot reproduce is an opinion with a number attached, and should be replaced by the rule itself.

Input is a tested dashboard and a critique record; the transformation is revision plus composition; the output is the pair of artifacts the December 2 meeting actually runs on. Note that the memo is written last and read first, which is the normal relationship between the two, and note that its first sentence is the sentence the whole chapter has been

building toward — the one the analyst would say if she had thirty seconds, which Section 12.18.5 asked students to draft and promised would be rebuilt around. This is the rebuilding.

VERIFICATION CHECK

Before you run: state, in writing, how many sentences the artifact contains that make a claim — every title, subtitle, annotation, source-band line, and memo sentence. That count is the denominator for C1.

After you run: run the complete audit before either artifact leaves your hands — the three transferable points of Table 8.5, all three of Table 12.5 on every view, and all three of Table 13.6 on the artifact as a whole. C1 is run sentence by sentence, and the count is the deliverable: report how many sentences you checked and how many you changed. C2 is already run, in Part A, and the record of it is the evidence. C3 requires the disclosure list: every simplification, including each filter's scope, with a line stating whether it is material. Then run the final check, which is the one that matters most and takes the longest to accept. Hand the memo alone, without the dashboard, to a third person, and ask what decision it recommends and on what basis.

If she can state both correctly, the memo works and the dashboard is what makes it persuasive.

Investigate if: she cannot state the recommendation, which means the dashboard has been carrying an argument the analyst never actually wrote down — and an argument that does not exist independently of the picture will not survive the first person who reproduces the picture differently; she states a recommendation stronger than the one you wrote, which means the memo's conditions are buried behind its first sentence; or she asks what the interval means, which means the store-equivalent conversion did not travel out of the annotation and into the prose.

13.13 Marketing Interpretation and Managerial Insight

The labs produced a tested dashboard, a critique record, and a one-page memo. On December 2 the chief executive will look at the screen, form an impression, ask three or four questions, and decide. This section translates the artifact into the decision language it has to survive in, and — as in every chapter since Part II — it does so partly by exhibiting the wrong readings, because a communication artifact attracts a family of misinterpretation that differs in an important way from the ones Chapter 12 cataloged. Chapter 12's wrong readings came from two sources: charts that misencoded, like the truncated axis in the circulating deck, and charts that encoded honestly while inviting the wrong comparison, like the calendar-time facet that made the Wave 4 stores look weak. This chapter's come from a third place — charts that encode correctly, invite the right comparison, and are then asked to carry more weight than a picture can bear.

The first wrong reading is the chapter's signature and will arrive within ninety seconds of the screen going up. "This dashboard says we should sign the leases." It does not, and the reason is not that a dashboard is incapable of carrying a recommendation — this one carries a decision sentence in its title band, and a callout stating a decision rule would be a legitimate and often excellent addition. The reason is that the seven views themselves are a display: four are favorable to the expansion, two are unfavorable, and one is a range, and no arrangement of them constitutes the conditional sentence the evidence actually supports. What the room is doing is inferring a recommendation from emphasis, and emphasis is easily read as endorsement, particularly by an audience that wants a decision and will accept help forming one. The correction is not to weaken the design. It is to state the recommendation out loud, in the analyst's own voice, with its conditions attached, so that the room is responding to a claim somebody made rather than to an inference nobody is accountable for. The general form of the lesson is worth carrying into every meeting a student will ever present in: *charts and marks do not automatically constitute a durable recommendation, and when nobody states one, the audience infers one from the layout and attributes it to the analyst anyway.* Given that the attribution happens regardless, the only question is whether the analyst controls what is attributed to her —

which is why a decision-support artifact pairs its evidence with explicit recommendation text and documented conditions.

The second wrong reading is the mirror of the first and is more expensive because it is quieter. "The analyst isn't sure, so this is really a judgment call for us." This is what an audience concludes when the uncertainty is presented and the recommendation is not, and it is the specific failure the concept box in Section 13.9 was written against. Note what has happened when it occurs: the analyst has done the harder and more honest thing — she has shown the interval, named the weak store, put the 24 percent repeat share on the screen — and has been penalized for it, because the room has no way to distinguish *calibrated* from *unconfident*. The correction is structural rather than rhetorical. The four-part shape — recommendation, evidence, uncertainty, reversal conditions — puts the recommendation first specifically so that the uncertainty that follows is heard as qualification of a stated position rather than as the absence of one. An analyst who leads with the interval will be heard as having nothing to say, no matter how good the interval is.

The third wrong reading belongs to the diagnostic band and is the one that most often causes real damage in real meetings. "Scottsdale is underperforming — let's look at what's wrong there." The dashboard shows one Wave 4 store below the range its siblings occupy, which is a fact, and the room has converted it into a project. Two things are wrong with the conversion. The first is inferential. With eight new stores and only four predecessor paths, one visibly low store is not, by itself, evidence of a systematic operational failure — nor, and this is the part students get backwards, is it something the sample size *guarantees*. The predecessor range is the observed minimum-to-maximum spread of four stores rather than a statistical prediction interval, and how often a store should be expected to fall outside it depends on the underlying distribution, on the dependence between successive weeks, on how different the store formats are from one another, and on the precise rule used to classify a store as below. None of those are known here, so no probability can be quoted, and a claim that one low store was "close to certain" is exactly the kind of confident-sounding sentence C1 exists to catch. What can be said is the useful thing: Scottsdale should be treated as a diagnostic case and monitored against a rule declared in advance, rather than selected after inspection and subjected to an open-ended search for explanations — which is the post hoc hunting Section 11.8 named in a different costume. The second problem is decisional: the meeting's question is whether to sign eight *new* leases, and Scottsdale's performance is evidence about the *selection* of sites rather than about the operation of that store. The correction is to give the store its proper role, which is neither exhibit nor project: it is the case that defines the recommendation's condition. The recommendation is not "sign eight" or "sign seven"; it is "sign the sites matched to comparables that reached coverage, and hold both the site matched to Scottsdale and the two with no qualifying comparable until the ramp evidence separates" — at which point the outlier has done useful work and has not become an operational distraction that consumes a quarter.

A fourth reading is worth naming even though it is not exactly wrong, because it is the one that most often ends a meeting badly. "Can you add the online channel to this?" Some version of it will be asked, and the honest answer is usually yes, in about twenty minutes. The failure is to answer it in the room, live, in front of the dashboard, which is what the interactivity makes tempting and what the deliberate absence of any filter control on the screen was designed to prevent. Building a new view during a decision meeting produces an unaudited chart on a projector in front of the decision-maker, which is every mechanic in Chapter 12 waiting to happen, and it converts a prepared argument into an improvisation. The professional response is to write the question down, answer it by end of day, and return the meeting to the decision — and the reason to name this in a textbook is that students taught to build fast are strongly inclined to demonstrate that they can.

The memo that accompanies the artifact has a fixed anatomy, and it is worth setting out in full because the exercises ask for it and because it is the shape of nearly every recommendation a marketing analyst will write for the rest of her career. It opens with the recommendation, which is a rule before it is a count: *sign the sites matched to comparables that reached four-week-average coverage; hold the site matched to an unsuccessful comparable and the two sites with no qualifying comparable; revisit all three in eight weeks, which the letters of intent make possible.* Applied to `chapter13_prospective_sites` under the two rules declared in Lab 13.1 Part A step 6, all eight sites are eligible — every one is suburban or resort — and six of them match an existing store: five to comparables that reached coverage, and one to Scottsdale, which has tracked below the predecessor range since week six. The remaining two match nothing, because no existing store shares their trade area's density and income

bands. That distinction is not cosmetic — one site has unfavorable evidence and two have none, and the eight-week review resolves them by different means. The rule therefore signs five and holds three, and the count is stated only alongside the rule and the table that produces it. It deliberately appears nowhere in an assertion title, because no view on the dashboard displays the prospective sites and C1 of Table 13.6 does not make exceptions for the analyst's own recommendation. It states the evidence in decision units and by view: each of the four mature suburban and resort stores reached a four-week average at or above the weekly coverage threshold in weeks 43, 46, 51, and 57 respectively, which is the only claim the coverage view supports — four stores are four stores, no lease term or required ramp period has been established against which to call that range fast or slow, and the milestone says nothing about whether any of them stayed above the line afterward. The memo states all three limits rather than leaving them to be inferred; seven of the eight Wave 4 stores are within or above the range those four occupied at the same store age — two above it and five inside it — so the newest cohort's apparent weakness is consistent with store age rather than with performance; and the suburban customer's $156 order against a chain average of $108 is a basket-composition fact rather than a pricing one, which provides a testable expectation for markets with comparable customer profiles but does not guarantee that the pattern will reproduce. It states the uncertainty: the FY2027 outlook is $1.25 million of chain contribution, of which $0.51 million is already banked and $0.74 million is a central estimate for the seven months from December, with the part still open running $0.62 million to $0.86 million — about $240,000 end to end, worth roughly 1.8 mature stores' annual contribution or ±0.9 stores around the estimate; the $92,000 footprint is a finance planning assumption rather than a measured quantity, and every coverage figure moves with it; every store-level figure on the screen describes the suburban and resort formats and the two chain-level figures are labeled as such, while the pipeline's flagship sites carry no footprint, no margin, and no predecessor range, so nothing here bears on them; and the format's repeat-order share of 24 percent against the chain's 47 percent is reflected in the contribution observed to date but remains a forward-looking acquisition-cost and durability risk that a second wave of stores would compound if it does not improve. And it states the reversal conditions: if the eight Wave 4 stores have not, by their store-age week 25 — the last week the predecessor range covers, and eight weeks from now, which is why the option period was negotiated to that length — held their position within or above that range, let the option lapse on the three held sites rather than executing; if the finance partner's footprint figure moves by more than ten percent when the leases price, every coverage week on the dashboard is recomputed before anything else is signed; when the transaction extract next refreshes, the model is refit on the months it has not yet seen and the outlook is rebuilt, and the recommendation is reopened if December lands outside the central 80 percent range for the month; and if suburban repeat-order share has not moved by the end of Q2, the expansion case becomes a customer-acquisition case rather than a real-estate one and belongs in a different meeting.

One property of that memo deserves a closing paragraph, because it is what makes this chapter's deliverable different in kind from Chapter 12's. Every artifact in Part II was valuable for the number inside it, and Chapter 12's pre-read was valuable for the errors it prevented. This chapter's artifact is valuable for a decision it makes *reversible*. The reversal conditions are the memo's most important sentences and the ones a junior analyst is most likely to omit, because they read like hedging and are the opposite: they are the mechanism by which a recommendation made under uncertainty on a Friday can be unmade in eight weeks without anybody having to admit to a mistake. Organizations that make decisions well are not organizations that are right more often. They are organizations in which being wrong is detectable and cheap, and the analyst who writes the detection criterion into the memo has done more for the quality of the decision than the one who got the point estimate closer.

13.14 Business Analytics in Practice

The chapter has practiced communication on StyleCraft. This section steps out of the fiction and into how visual communication actually functions inside organizations — where a single screen can acquire institutional weight nobody intended, where an entire company decided that the deck was the problem, and where a redesign produced measurable operational change that had nothing to do with aesthetics.

The first vignette is the daily flash and the governance burden that follows it. It is a composite drawn from recurring organizational practice rather than a report on one named firm, and the pattern recurs across retail, media,

and consumer businesses: an analyst builds a simple morning dashboard — yesterday's revenue, orders, conversion, and a comparison to plan and to last year — for a specific manager who asked for it. It is useful. It circulates. Within a quarter it is the first thing the chief executive opens, and within two it is the number the company uses in conversation. What follows is a set of obligations that the original builder did not sign up for and that most analytics teams learn about the hard way. The flash must now be right by 7:00 a.m. every day, which makes it an operational system with an on-call expectation rather than a workbook. Its definitions become the company's definitions, so that a change to how returns are netted or how a channel is classified is no longer a data decision but a communication event requiring notice, because executives who have watched a number every morning for a year detect a two-point discontinuity instantly and interpret it as a business event. Its availability becomes a dependency, so that a failed extract at 6:40 a.m. is a leadership problem. And its scope is under constant pressure, because every executive who reads it wants one more tile on it, and a screen designed around a three-second read degrades one reasonable request at a time into the undifferentiated grid this chapter opened with. The teams that handle this well institute two practices that connect directly to Part III. They version the definitions and announce changes in advance, which is Section 3.8's data dictionary promoted into a communication protocol. And they defend the hierarchy explicitly, treating a request to add a tile as a request to remove one, which is Section 13.7's discipline given organizational teeth. The practice lesson for a new analyst is that a widely read artifact is infrastructure, and that the moment to negotiate its ownership, its refresh guarantee, and its change process is before it becomes the chief executive's first screen rather than after.

The second vignette is the memo-first culture, and it is the clearest institutional expression of this chapter's argument that a screen alone does not carry a recommendation. Amazon is the best-documented example: the company's 2017 shareholder letter states plainly that it does not use PowerPoint in its senior meetings, that it uses narratively structured six-page memos instead, and that those meetings begin with everyone reading the memo silently (Bezos, 2018). The mechanism is worth understanding rather than admiring, because the practice spread to firms whose meetings it did not fit. What the memo does is force the argument into prose, where the connective tissue — *therefore*, *because*, *unless* — has to be written out and can be inspected. A deck lets a presenter place two charts on consecutive slides and let the audience supply the causal link, which is efficient when the link is sound and is the most common way an unsupported conclusion enters a decision when it is not. What the memo does not do is replace visual evidence; the memos contain charts, and appendices of them. The practice lesson for a marketing analyst is not that decks are bad. It is that if the recommendation cannot be written as a paragraph with its connectives intact, the analyst does not yet have a recommendation — she has a sequence of true statements and a hope that the audience will assemble them the way she would. That is exactly what Section 13.12.7's final verification check tests when it hands the memo to a third reader without the dashboard.

The third vignette is the redesign that produced a measured outcome, and it is worth telling because it is the vignette in which visualization work is evaluated the way the rest of the business is. It is likewise a composite. A campaign-performance dashboard exists, is opened daily by a dozen channel managers, and takes each of them roughly ten minutes to work through because the KPI hierarchy is flat and the pacing decision they actually make — whether to move budget between channels today — requires them to read four separate tiles and do arithmetic. A redesign restructures the same data around that decision: one primary view showing each channel's spend against its pacing target with the variance in dollars, the supporting detail one click away, and the eleven tiles that nobody used removed entirely. The reported effects, where teams have bothered to measure them, are consistent in shape if not in magnitude — a substantial reduction in time-to-decision, a measurable increase in how often the pacing action is actually taken, and a decrease in the number of clarifying questions sent to the analytics team, which is the effect most visible to the analysts themselves. What did not change was the data, the model behind the pacing target, or the accuracy of anything. The practice lesson is that the return on communication work is real, is measurable by the operational behavior it changes rather than by anyone's opinion of the design, and is almost entirely invisible in the metrics analytics teams usually report about themselves. An analyst who wants credit for it has to instrument it, which means recording the decision the artifact is meant to support and checking, later, whether that decision got made faster or better.

In your first analyst job, these three compress into one expectation, and it is this chapter's closing thread: analytics careers compound through communication, because the last mile is the visible mile. The technical work of

Chapters 4 through 11 is what makes an analyst employable, and it is also the part of the work that nobody outside the analytics team observes. What colleagues and managers observe is the artifact, the meeting, the memo, and the answer given to the awkward question — and their model of an analyst's competence is built almost entirely from those. This is not a complaint about superficiality; it is a description of how organizations necessarily allocate trust when they cannot inspect the work directly. The practical consequence is that the skills in this chapter compound in a way the technical ones do not. An analyst who is known to produce artifacts that can be read in three seconds and defended for thirty minutes gets invited to the meetings where decisions are made, which is where she learns what decisions actually require, which makes her next analysis better aimed. The analyst whose work is correct and unreadable is asked for numbers, and being asked for numbers is a career of answering other people's questions. The difference between the two paths is not intelligence and is not usually effort. It is whether the last two feet were treated as part of the job.

13.15 Ethics, Persuasion and Distortion

Section 13.14 ended on the observation that communication is what an analyst is judged by. This section takes up what follows from it, and it is the chapter's hardest material because it is the place where this guide's running ethics discussion — data use (Section 1.13), problem framing (Section 2.12), measurement design (Section 3.13), cleaning as editorial power (Section 4.14), honest summarization (Section 5.14), differential treatment of segments (Section 6.16), causal language (Section 7.15), acting on predictions about people (Section 8.15), fairness across groups (Section 9.15), forecast accountability (Section 10.15), experimentation on customers (Section 11.15), and misleading chart mechanics (Section 12.16) — arrives at a boundary that cannot be drawn by rule.

Chapter 12's ethics were, by comparison, easy. The five mechanics it cataloged are wrong in every context: a truncated bar axis misstates a proportion whether the analyst intended it or not, and the obligation it creates is to check. This chapter has no such list, because every technique in it is legitimate and every one of them is also the mechanism by which a reader can be steered. Placing the cost-coverage view at the upper left in the largest frame is the hierarchy of Section 13.4 doing its job; it is also a decision that the reader will encounter the expansion's best evidence first. Coloring the four stores that reached the four-week-average coverage milestone in an accent hue and the rest in gray is the emphasis discipline of Section 13.7; it is also a decision about which stores the reader will look at. Writing "seven of the eight new stores are within or above the range their predecessors occupied" is an assertion title verified against its view; the same view honestly supports "one of the eight new stores has tracked below its predecessors since week six," and choosing the first sentence is choosing what the reader takes away. None of these misstates a quantity. All of them shape a conclusion. Hullman and Diakopoulos (2011) gave the general phenomenon a name and a taxonomy, cataloging the framing effects available in narrative visualization — what is included, what is omitted, how a view is annotated, what order the views arrive in — and observing that these rhetorical choices are unavoidable rather than optional, since every artifact makes them and an artifact that declined to make them would not be readable. That is what makes the chapter's ethics a matter of judgment rather than of checking, and it is why the boundary has to be argued rather than listed.

CONCEPT

Emphasis Directs Attention; Distortion Directs Belief

The working distinction this guide adopts is between techniques that change what a reader *looks at* and techniques that change what a reader *concludes is true of the data*. Emphasis, ordering, sizing, decluttering, and titling all direct attention: they make one true thing easier to find than another true thing, and a reader who spent an hour with the underlying evidence would arrive at the same understanding of the numbers, more slowly. Distortion changes the understanding itself: after encountering it, a reader believes something about the quantities that the data does not support, and would revise that belief on seeing the evidence. Chapter 12's mechanics are distortion by construction.

> But distortion is available in this chapter too, without any mechanic, through three moves that are worth naming because they are the ones that occur. Selection: showing the four views that favor a conclusion and omitting the three that qualify it, so that the reader's picture of the evidence is not the analyst's. Framing: presenting a quantity in the form that produces the intended reaction — "repeat share improved 20 percent" for a move from 20 to 24 percent — where the alternative framing is equally true and produces a different one (Tversky & Kahneman, 1981). Aggregation: grouping the categories whose individual values would change the ranking into an "other" that hides them.
>
> Each of these produces a false belief with true components, which is why the test for them cannot be *is every element accurate* and must instead be the counterfactual: *would this reader revise her conclusion if she saw everything I saw?* If the answer is yes, the artifact is distorting regardless of how carefully each of its elements was verified.

That counterfactual is the standard, and three practical obligations follow from it.

The first is the obligation of the omitted view, and it is the one this chapter's technique makes most acute. Section 13.7 requires decluttering, and decluttering requires removal; the go/no-go dashboard came down from the pre-read's eight views to seven, three of them cut and two purpose-built ones added, and from seven filter controls to none, and each removal was defended in terms of the reader's ten minutes. The obligation is to sort the removals into two piles. A view removed because it is redundant, or because the memo carries it better, or because it answers a question this reader will not ask, is a legitimate simplification. A view removed because it complicates the recommendation is not, and the fact that the same sentence — "it was cluttering the screen" — is available to describe both is precisely why the sorting must be done deliberately and in writing, which is what C3 of Table 13.6 exists to force. The practical test is the one the labs run: list every removal, and for each state whether the reader's decision would change if she saw it. Anything that survives that question with a yes goes back on the canvas, at whatever size, however inconvenient.

The second is the obligation of the disclosed state, and it is narrower and almost mechanical. An interactive artifact is in some state when the reader encounters it, and that state was chosen by the analyst. A filter defaulting to the last six months, a store type excluded because it "isn't comparable," a forecast whose origin is months before the meeting, a filter scoped to three worksheets rather than seven — each may be entirely correct and each is invisible to a reader who has no reason to check the controls. The standard is that the state must be stated in words on the canvas, not merely reflected in a control, because a control reading "Suburban, Resort" is a fact about an interface and a caption reading "Physical stores only; online orders excluded; suburban and resort formats only; July–November reported, December–June forecast from a model fit through June 30; no filters applied" is a fact about the evidence. The go/no-go dashboard makes this obligation sharper by carrying no controls at all, which removes the interface from the question and leaves nothing but the words: a screen with nothing to adjust says nothing whatever about what it contains unless the source band says it. This is the footer discipline of Section 12.9 carried into a medium where the artifact can change under the reader, and it is the reason the source band survived the decluttering that removed almost everything else.

The third is the obligation that comes with an audience that cannot check, and it is the one that grows over a career. Chapter 12's readers were colleagues with the extract and a week. This chapter's reader is a chief executive who will never see the file, and the readers of the artifacts these students will build in their first jobs are further away still — a board, a regulator, a customer, a journalist reproducing a chart out of a press release. Every step in that direction transfers responsibility from the reader to the analyst, and the transfer is complete when the artifact travels beyond the room where its author can be questioned. The working standard is therefore that a persuasive artifact must be built so that it would survive its own audience being fully informed: the analyst should be able to imagine handing the chief executive the entire pre-read, the certified extract, and an afternoon, and to expect that the chief executive would reach the same decision. An artifact that depends on the reader not having that afternoon is not a communication artifact. It is an argument that requires an ignorant audience, and the fact that the audience is in fact busy rather than ignorant does not change what the artifact is doing.

One boundary closes the section and the guide's ethical thread. There is a version of this material that collapses into paralysis — every design choice shapes a conclusion, therefore every design choice is suspect, therefore the honest analyst makes no choices and produces the undifferentiated grid of Section 13.1. That conclusion is wrong and is worth rejecting explicitly, because it is where careful students land. A screen with no hierarchy is not neutral; it is a screen whose hierarchy was set by the order the views were built in, which serves nobody and which still produces a conclusion in the reader's head, just an accidental one. Refusing to choose is itself a choice with consequences, and it is a worse one, because it is unaccountable. The analyst's position is not that she should avoid influencing the decision — influencing the decision is the job, and it is why the organization employs her. It is that she should influence it with the evidence, in the direction the evidence points, having first checked that it points there, and having built the artifact so that a reader who went and looked would agree. That is a demanding standard and it is achievable, and the difference between meeting it and not is usually a written list of what was left out.

13.16 Chapter Summary

This chapter completed Part III's account of visualization by taking its second job, and it began by insisting that the second job is not a polished version of the first. An analytical artifact is built for a reader who has time, stake, and a reason to interrogate; a communication artifact is built for a reader who has a decision, a deadline, and no prior exposure to the evidence, and the design constraint is therefore the reader rather than the data. Audience analysis makes that constraint writable — who reads, what decision, how long, what she knows, what would change her mind — and the course's three-second reading test makes it measurable, by isolating what an artifact communicates before a cooperative reader decides to work at it. From there the chapter chose the object before designing it: exploratory artifacts let a reader investigate by her own path and explanatory ones deliver a finding along a path the author chose, and the go/no-go artifact turned out to be a data story delivered on a dashboard's canvas with a memo attached, which settled half a dozen later decisions including why its filter panel shrank from seven controls to none. The screen was then given a structure. The KPI hierarchy spatialized the metric discipline of Section 3.5 into primary, supporting, and diagnostic tiers, so that the two views the lease decision actually turns on — the store-age week in which a mature store's trailing four-week average of contribution first reached the $1,769.23 weekly share of a declared $92,000 annual footprint, and where the eight Wave 4 stores sit against the range their predecessors occupied — occupy the largest frames on the reader's entry path, and so that the case's two weakest facts sit on the canvas rather than in a backup slide. Layout converted the hierarchy into geometry through reading order, grouping, density, and whitespace, on the principle that position is a claim about importance whether or not the analyst intended to make one.

The chapter then supplied the techniques Chapter 12 deliberately withheld. A narrative arc organized the views around the finance partner's objection rather than around the strongest evidence, on the grounds that an audience holding an unaddressed doubt is not persuaded by anything else. The assertion title converted each view from a display of a quantity into a statement of a finding, at the cost of an obligation that a label title does not carry — that the sentence be verifiable from the marks beneath it, in the view's own units, with a verb the evidence hierarchy of Section 11.10 licenses and no number the view does not contain. Preattentive attributes supplied the emphasis toolkit, with the conjunction penalty and the conditions under which the effect weakens explaining why emphasis is a budget rather than a technique, and data-ink discipline supplied the subtraction that gives emphasis something to work against — a subtraction the chapter framed honestly as deleting one's own work and made survivable by moving rather than destroying. Interactivity was bounded to navigation and communication, with the rules that every finding must be visible in the default state, that a highlight action can only mark entities its target views actually contain and should therefore run from the smaller value set to the larger, that a filter's scope must be chosen and disclosed rather than applied to everything — and that a control comes off entirely when no safe scope exists, which is what happened to both of this artifact's surviving filters and is why the finished screen carries none, declaring its fixed state in the source band instead — and that a title over a parameterized view must be built to stay true in every state the control can produce. Uncertainty then received its own treatment, because Chapter 10's interval and Chapter 11's break-even had to survive a room that wants a number: stated as a central estimate with a central 80 percent range rather than as a most-likely value and a range that cannot be ruled out; structured as actual-to-date plus

forecast-to-go, so that the five months of the planning year that had already happened by December 2 sat outside the band as history rather than inside it as risk; stated in contribution rather than revenue, so that the store-equivalent conversion needed no margin and carried no unestablished assumption; converted into decision units in both framings — 1.8 mature stores end to end, about ±0.9 around the estimate; and drawn as regions whose outlines are described upstream as ordered vertices and drawn as polygons, because a reference band renders at constant height, a prediction interval does not, and two bound measures dropped side by side on Rows do not stack into a region on their own. The four-part shape — recommendation, evidence, uncertainty, reversal conditions — was the chapter's answer to the false choice between false confidence and useless hedging, and its fourth part is what turns an interval into a monitoring plan.

The last movement closed the loop from artifact to decision. A screen displays evidence and can carry a recommendation only as text somebody wrote, so it is paired with a one-page memo that states the recommendation, is written before the meeting because a recommendation that will not survive being written will not survive being questioned, and carries the reversal conditions that make the decision cheap to unmake. The usability check then supplied this chapter's verification theme, which is the first in the book that the analyst cannot run alone: predict what a fresh reader will say, show the artifact for three seconds, record the answer verbatim, say nothing for ninety seconds more, and grade the predictions — with a mismatch in the subject of her sentence diagnosed as a hierarchy failure and a mismatch in the verb as a title or emphasis failure, and with the whole exercise reported as the formative check it is rather than as validation. The critique protocol made the same exchange survivable between peers by fixing its order. The AI section named the failure modes that follow from the assistant drafting the interpretation rather than the instrument — the overclaiming title in its three shapes, the confident recommendation that deletes the finding and keeps the tone, the uncertainty that disappears when a paragraph is made punchier, the layout that reflects convention rather than hierarchy — and installed the routine that requires every candidate sentence to arrive with its own verification line. And the ethics section drew the line Chapter 12 declined to draw: emphasis directs what a reader looks at, distortion directs what she concludes is true, and the test between them is a counterfactual rather than a checklist — would this reader revise her conclusion if she saw everything I saw? The obligations that follow are the sorted list of what was removed, the filter state and scope stated in words rather than shown in a control, and the standard that a persuasive artifact must be built to survive its own audience being fully informed. What the section explicitly refused is the paralysis that the argument invites, since a screen with no hierarchy is not a neutral screen but one whose hierarchy was set by accident.

StyleCraft's storyline closes here. The decision gets made on December 2 and the eight leases price on Friday, which ends a thread that has run since Chapter 2 arrived with a vague request to look at the expansion. It is worth noticing what carried it. Not the model — the forecast is one of seven views and the smallest claim on the screen. What carried it was a chain of small verified moves: a metric defined at the grain the question required, a baseline the result had to be read against, a threshold declared before the results were inspected, a scaffold that refused to let an absent week disappear, a coverage rule that asked for four weeks rather than one, an interval placed only around the part of the year that was still open, a sentence checked against the marks beneath it, and a reader who was asked what she saw instead of told what she should have seen. That chain is the guide's actual subject, and the dashboard is only the last link of it.

What remains is yours to build. Your instructor may release the Project #2 brief with this chapter, and its deliverable is the artifact this chapter teaches: an executive-facing dashboard or visual report, paired with a written recommendation, tested on a reader before it is submitted. Three appendices carry the instruments you will need and are worth opening before you start rather than after. Appendix D holds the AI-Use Documentation Template, which is what makes an AI-assisted artifact auditable rather than merely fast. Appendix E holds the Data Visualization Checklist, which is where Chapter 12's mechanics and this chapter's three communication points live as something you can run rather than remember. Appendix F holds the project rubrics, including Project #2's, which you should read before you open a canvas, because a rubric read afterward is a grading instrument and a rubric read beforehand is a specification.

And the standard is the one Chapter 1 set. AI may draft the chart, the title, the layout, and the paragraph; the student remains the analyst of record. Specify what the artifact must do and for whom, before building it. Predict what it will show and what a reader will take from it, then verify both — the first against the marks, the second

against a person. Explain every choice you made, including the ones the software made for you and you accepted. And document the exchange, so that the next reader of your work can see not only what you concluded but how you came to be entitled to conclude it. That workflow was introduced in Chapter 1 as a way to use a new set of tools responsibly. Thirteen chapters later it is simply what it means to do the work.

13.17 Exercises for Practice and Homework

The following exercises practice the chapter's main habits: state the reader before the design, name the decision the artifact must support, put the primary view where the eye lands first, write the sentence you want the reader to leave holding and then verify it against the marks, remove what does not help her believe it, show the uncertainty in units she cares about, and find out from an actual person what your artifact says. Not everything below is required. Core practice, which every student should complete: 13.17.1, 13.17.2, 13.17.3, and 13.17.4. Homework options, from which your instructor will assign a subset: 13.17.5 and 13.17.6. Instructor-led discussion, prepared for class rather than submitted: 13.17.7 and 13.17.8. Each exercise also carries its label individually. Your instructor may assign selected exercises as preparation for Project #2; consult the course schedule for deadlines. Exercise 13.17.5 produces the artifact the project is built on, and the Project #2 rubric in Appendix F should be read before starting it rather than after.

13.17.1 Concept Check (Required Practice)

Answer each in two or three sentences, in your own words.

- State the five questions an audience analysis must answer, and for each, name one design decision it determines.
- Explain what the three-second test measures and what it does not, state why this guide uses three seconds rather than the five used in first-impression testing generally, and state why the correct response to a failed test is to revise the artifact rather than the reader.
- Distinguish an exploratory artifact from an explanatory one, and give one example of an artifact built for an external audience that is nonetheless exploratory.
- Define a KPI hierarchy, and explain why a canvas of six equally sized views has communicated a claim the analyst does not believe.
- Explain what preattentive processing is, name two conditions under which the effect weakens, and state why a dashboard that emphasizes six elements has emphasized none.
- Distinguish data-ink from non-data-ink, and name two elements of non-data-ink that should nonetheless never be removed, with the reason for each.
- State the obligation an assertion title carries that a label title does not, and give one example of a title that is true but not earned by the view beneath it.
- Explain why a Tableau reference band cannot draw a prediction interval that widens with the horizon, why placing the lower and upper bound measures side by side on Rows does not produce a filled region between them, and what must be true of the underlying data before the interval can be drawn correctly.
- Define first observed weekly coverage, state why it is not a payback period, and explain why a complete store-week scaffold does not by itself move the first qualifying week later while it is indispensable to the four-week variant of the same rule.
- Explain the difference between emphasis and distortion as this chapter defines them, and state the counterfactual test that separates them.

13.17.2 Audience First (Required Practice)

For each of the following deliverables, write the five-line audience analysis of Section 13.2, then choose an artifact from Table 13.1 and justify the choice in one sentence, then state the intended three-second takeaway.

- A weekly summary of email campaign performance for four channel managers who will use it to decide where to move next week's sends.

- A one-time analysis, requested by a chief financial officer, of whether the loyalty program's discount cost is justified by the incremental revenue it produces.
- A monitoring artifact for a merchandising team that needs to notice, within a day, when a product category's sell-through falls behind plan.
- A summary of Chapter 11's A/B test result for a marketing leadership team deciding whether to roll the tested creative out to the full list.

Then answer the question the exercise is actually about. Two of the four are cases in which a dashboard is the wrong answer. Name them, state what the right artifact is, and explain what specifically about the reader or the decision rules the dashboard out.

13.17.3 Titles That Earn Their Verbs (Required Practice)

Each of the following pairs a view with a proposed title. For each, state whether the title is earned by the view, and if it is not, name which of the three overclaim shapes it exhibits — causal or predictive verb, generalized population, or imported number — and rewrite it as a sentence the view supports.

- A sorted bar chart of average order value by home metro, at order grain, with the chain average drawn. Title: "Suburban customers drive StyleCraft's growth."
- A line chart of the four-week average of weekly contribution by store age for four stores, on a complete store-week scaffold, with a reference line at $1,769.23. Title: "The suburban format pays back in under a year."
- A 100 percent stacked bar of category revenue share by metro. Title: "Occasionwear is roughly three times the share of suburban revenue that it is of urban."
- A fiscal-year contribution line, actual through November and forecast thereafter, with 80 and 95 percent interval regions on the forecast months only. Title: "Contribution will reach $1.25 million this year."
- A bar chart of repeat-order share by store type with the chain share drawn. Title: "Suburban repeat-order share is 24 percent against a chain average of 47 percent."
- A scatterplot of items per order against occasionwear share, one mark per store. Title: "Bigger baskets are an occasionwear story."

For the second item, note that all three overclaim shapes are present at once, which is why it is the most instructive title in the list: the verb makes a claim about the future, the noun names a quantity the view does not measure, and the sentence generalizes four observed stores to a format — while also being false of the one that took fifty-seven weeks. State all three, and explain in one sentence why the middle error is the hardest of them to notice.

Then, for the two titles you judged earned, write the alternative title that the same view *also* supports and that would leave a reader with a different impression, and state in one sentence how you would choose between them and what makes that choice a matter for Section 13.15 rather than for Section 13.6.

13.17.4 Subtract, Then Emphasize (Required Practice)

Take any one view you built for Chapter 12's pre-read and convert it into a communication view without changing a single underlying query.

First, write the assertion title. Then list, before touching anything, every element you intend to remove and every element you intend to emphasize, with the reason for each stated in terms of that title. Then make the changes, and record any element you found yourself removing that was not on your list, with a note on why you had not anticipated it.

Finally, produce the before-and-after pair as images and write two sentences: one stating what changed in the reading, in the form Section 12.13.5 established, and one stating what the analytical version could show that the communication version cannot. The second sentence is the point of the exercise. If you cannot name anything that was lost, either the analytical view was overbuilt or the communication view has not been simplified enough, and both are worth discovering.

13.17.5 The Go/No-Go Dashboard and Memo (Homework Submission)

Build the artifact described in Section 13.12, as a Tableau workbook plus a one-page memo. This is the deliverable Project #2 builds on; read the rubric in Appendix F before beginning.

The workbook must contain: the written specification from Lab 13.1 Part A, including the reconciled Table 13.7, the sealed intended takeaway, and the sealed predictions, as a text object or an accompanying page; the store-week scaffold and the certified metrics file, with your own reproduction of every row of Table 13.7 and any mismatch reported as a finding; the seven views of Table 13.8, each rebuilt rather than duplicated, each carrying an assertion title and a grain subtitle; the assembled dashboard at a fixed size, built in containers, with no filter controls and its fixed state declared in words in the source band, one highlight action running from the smaller value set to the larger so that every selection it can produce has a match, one parameter used as navigation with a title that is true in every state it can produce, and alt text on every view; and, on a separate analytical tab, the views and the seven filter controls that came off the canvas.

Four construction requirements are graded specifically because they are the ones most often faked. The predecessor region in View 2 and the interval regions in View 5 must be built from bounds that vary along the horizontal axis, and you must demonstrate it: report the region's upper and lower vertices at three separated store-age weeks and at the first and last forecast months, and show that they differ. View 2's line layer must contain exactly the eight Wave 4 stores and the region must survive the restriction; report the line count and state how you filtered without deleting the band-outline rows. View 5 must carry a horizon-total block whose figures come from the outlook file's horizon rows, and no shading may appear left of the December forecast-to-go seam; report the block's figures beside the sum of the seven monthly bounds and state how far apart they are and why. And every weekly path must be built on the scaffold; report, for one store, `First Observed Coverage Week` and `First 4wk Coverage Week` computed with and without it, and state which of the two moved, which did not, and why the difference is what the scaffold is for.

The memo must contain the four parts of Section 13.10 — recommendation, evidence, uncertainty, reversal conditions — in that order, on one page. Its uncertainty paragraph must state the range still open in store-equivalents in both framings and show the division that produced each, including which denominator you chose and why, must name the $92,000 footprint as a finance assumption rather than a measurement, and must state the population every figure describes. Its appendix must list every view removed from the canvas with one line apiece on where it went and whether its absence is material to the decision, per C3 of Table 13.6.

Submit both, plus the completed audit: the three transferable points of Table 8.5, all three of Table 12.5 on every view, and all three of Table 13.6 on the artifact, with the C1 sentence count reported — how many sentences you checked and how many you changed.

13.17.6 AI Titles, Prose, and What Disappeared (Homework Submission)

Run the five-step routine of Section 13.11's AI in Practice box, and document it per Appendix D. Submit the following.

First, the title exercise: your request prompt for one view, the twelve candidates with the assistant's verification lines, your written prediction of how many would overclaim and in which shapes, your audit of each candidate against the built view, and your grade of the prediction including both misses and false alarms.

Second, the prose exercise, which is the more revealing of the two. Write your recommendation paragraph yourself, carrying the interval and the reversal conditions. Ask an assistant to make it clearer and more concise for a chief executive. Then diff the two versions clause by clause rather than reading the new one, and report exactly what was removed. State whether each removal is a legitimate compression or a deletion of a finding, and produce a final version that is shorter than yours and contains everything yours contained.

Third, one paragraph on the pattern. Across both exercises, what did the assistant reliably add, and what did it reliably remove? Name the mechanism you think produces each, and state one change to your own prompting that you will make as a result.

Your grade rests on the audit and the diff, not on the assistant's output. An exchange in which the assistant deleted your interval and you caught it is a better submission than one in which it produced a paragraph you accepted.

13.17.7 Where the Line Is (In-Class Discussion)

Each of the following describes a design decision an analyst might make on the go/no-go dashboard. For each, state whether it is emphasis or distortion under the standard of Section 13.15, name the obligation it triggers if any, and state what disclosure would make it acceptable if one exists.

- Placing the coverage view in the largest frame at the upper left, and the repeat-share view small at the bottom.
- Removing the repeat-share view entirely, on the grounds that the contribution already observed reflects it.
- Titling the ramp view "seven of the eight new stores are on track" rather than "one of the eight new stores has tracked below its predecessors since week six."
- Shipping the dashboard with no filter controls at all, and stating the population and the window only in small type in the source band.
- Reporting the coverage milestone as the first single week above the threshold rather than the first four-week average above it, in a memo recommending a ten-year lease.
- Describing the repeat-share movement from 20 to 24 percent as a 20 percent improvement.
- Grouping the eleven smallest product categories into "other" in the mix view, where two of them individually exceed the fourth-largest named category.
- Extending the outlook's horizontal axis six months past the end of the fiscal year so that the interval regions occupy a smaller share of the frame.

For the last item, state what would have to be true about the reader for the choice to matter, and connect your answer to the mechanic Chapter 12 owns that it most resembles.

13.17.8 Ten Minutes with the Chief Executive (In-Class Discussion)

You are the analyst in Section 13.1. It is 9:04 a.m. on December 2. The dashboard is on the screen, you have said your recommendation sentence, and the chief executive has responded: "The merchandising team's numbers last week said suburban revenue jumped last quarter. Yours says the four stores we can observe didn't get their four-week averages up to the weekly fixed-cost threshold until somewhere between weeks 43 and 57. Which is it?"

The numbers she is referring to are the ones from the deck Chapter 12 repaired: a real 4.6 percent quarterly increase, drawn on a truncated axis, built by a junior analyst on the merchandising team. She is not in the room — she was never a pre-read participant. Her manager, the head of merchandising, forwarded the deck with the note "this is great, let's use these," and is sitting at the table.

Prepare a position, and address each of the following explicitly. What do you say in the next thirty seconds, given that each statement is true of what it measures, that the two measure different quantities over different windows, that the discrepancy is partly an encoding artifact rather than a data disagreement, and that the person who forwarded the other chart — and who has argued the composition case for a year — is sitting at the table? Which of Chapter 12's mechanics do you name, and do you name it at all, or do you answer only about your own number? How do you keep the meeting on its decision rather than converting it into a reconciliation exercise, and what do you commit to doing after the meeting instead? What would you do differently if the other deck had come from the chief executive's own prior briefing rather than from a colleague's team? And what would change if the analyst who built it *were* in the room? And finally, a question with no comfortable answer: if you had not built the pre-read, and the only artifact in the room were the other deck, would this decision have gone differently — and what does your answer imply about the value of work whose only visible output is a decision that did not go wrong?

13.18 Glossary of Terms

This glossary includes only the terms introduced in this chapter. Each definition is tied to the sources used in the chapter rather than added for decoration.

Assertion title. A chart title that states the finding the view supports, written as a complete sentence with a subject and a verb, in contrast to a label title, which names the fields displayed; because the reader acquires the title before decoding the marks, an assertion title determines what the view is understood to show and must be verifiable from that view's marks in that view's units (adapted from Knaflic, 2015; Kosslyn, 2006).

Audience analysis. The explicit statement, made before an artifact is designed, of who will read it, what decision it must support, how long the reader will give it, what she already knows, and what evidence would change her mind; its output is a set of design constraints rather than a description (adapted from Knaflic, 2015; Few, 2006).

Critique protocol. A structured procedure for feedback on a visual artifact in which the reviewer reports the takeaway, asks the intent, identifies impeding elements, and only then proposes changes, while the author listens without defending; the fixed order is the mechanism, since reporting before learning the intent yields an uncontaminated measurement (adapted from Krug, 2014; Few, 2006).

Dashboard action. An interaction configured on a dashboard by which activity in one view changes another — filtering, highlighting, or navigating to a different sheet — used to link an entity across views or to move a reader from a summary to its supporting evidence on request. A highlight action operates on matching field values and does nothing in a target view that does not contain them; the communicative test is whether the action answers a question the analyst predicted (adapted from Heer & Shneiderman, 2012; Shneiderman, 1996; Tableau, 2026b).

Data-ink and decluttering. Data-ink is the portion of a chart's ink representing data values; non-data-ink is everything else. Decluttering is the systematic removal or de-emphasis of non-data-ink and duplicated data-ink, undertaken to raise the contrast between what carries meaning and what does not, and retaining the non-data-ink — reference lines, direct labels, annotation, provenance — that helps a reader interpret correctly (adapted from Tufte, 2001; Few, 2012).

Data story. An ordered sequence of views, each carrying a stated point, connected so that the sequence constitutes an argument, typically moving from an accepted situation through a complication to a resolution; its defining property is that the order is chosen by the author and is load-bearing (adapted from Segel & Heer, 2010; Knaflic, 2015).

Exploratory and explanatory artifacts. An exploratory artifact lets a reader investigate by her own path and supports questions the builder did not anticipate; an explanatory artifact delivers a specific finding along a path the builder chose. The distinction is separate from the analysis-communication distinction of Section 12.2, and the most common design error in business visualization is building an exploratory object for a reader who needed an explanatory one (adapted from Knaflic, 2015; Segel & Heer, 2010).

Weekly coverage, observed and four-week-average. First *observed* weekly coverage is the earliest store-age week in which a store's weekly contribution reaches the declared weekly share of its annual fixed footprint; first *four-week-average* coverage is the earliest week in which the trailing four-week average of that contribution reaches it. Both are coverage milestones rather than payback periods, since nothing accumulates, and neither establishes that the store remained above the threshold afterward — a four-week average can be lifted over the line by one strong week and fall back below it. The four-week-average measure is the decision rule in this chapter because it is the less easily gamed of the two, not because it certifies persistence, and the gap between the two is itself informative. Because store age is derived from the opening date, a complete store-week scaffold does not by itself move the observed week; it changes the drawn path and the comparison range, and it is a precondition for the four-week-average measure, whose window cannot be computed on an extract with holes in it (Section 13.12.1; Sections 2.7 and 12.9).

Interval ribbon. A shaded region drawn around a forecast or a comparison set whose upper and lower edges vary along the horizontal axis. Because a Tableau reference band is defined between two values at table, pane, or cell scope and therefore renders at constant height, and because separate continuous measures on Rows occupy separate axes rather than stacking into a region, the ribbon is described upstream as an ordered set of vertices — along the lower bound left to right, back along the upper bound — and drawn with the polygon mark type using the Path shelf. Nested intervals are emitted as edge-to-edge regions rather than layered ones, so that no drawing order can conceal one behind another, and the series line is brought over the regions with **Move Marks to Front** rather than by relying on which axis a pill sits on (Section 13.9; Tableau, 2026a, 2026h, 2026i, 2026j).

KPI hierarchy. The ordering of a deliverable's metrics into tiers by proximity to the decision — primary, supporting, and diagnostic — and the expression of that ordering in the artifact's visual prominence, so that size, position, and emphasis correspond to decision relevance; a hierarchy stated in the analyst's head but not in the layout is not communicated (adapted from Few, 2006, and the metric and KPI roles of Section 3.5).

Preattentive attributes. Visual properties processed in parallel across much of the visual field before focused attention is directed, so that an object differing from its neighbors in one such property can be detected rapidly and with little increase in search time as the number of neighbors grows. The effect is strongest with a single distinctive feature and homogeneous distractors, and degrades with target-distractor similarity, heterogeneous displays, conjunctions of two attributes, and displays in which several attributes vary at once (adapted from Treisman & Gelade, 1980; Healey & Enns, 2012).

Three-second test. A course heuristic: showing a finished artifact to a representative reader for approximately three seconds, removing it, and asking what it was about. It measures what the artifact communicates before the reader decides to work at it, and a failure is a defect in the artifact rather than in the reader. The formal first-impression procedure in the usability literature is normally run at five seconds (Doncaster, 2014); three is used here as a deliberately demanding executive-dashboard constraint (course heuristic; informed by Doncaster, 2014; Few, 2006; Krug, 2014).

Usability test. A structured observation in which a representative reader who has not seen the artifact is given a task and observed while attempting it, with the analyst recording what she looks at, says, asks, and concludes, and refraining from explaining or guiding; a hesitation or misreading is evidence of a possible mismatch among artifact, task, and audience and is investigated as a candidate defect in the artifact. A reader who has seen an artifact can no longer supply a fresh first impression of it, though she remains available for later task-completion testing (adapted from Krug, 2014; Nielsen & Landauer, 1993).

13.19 Further Readings

Students who want additional background may begin with the following readings. The communication treatments are listed first, the perceptual and experimental evidence second, and the testing practice and tool reference last.

- Knaflic (2015) for the most direct practitioner's treatment of the material in Sections 13.6 and 13.7 — audience, story, decluttering, and emphasis, worked through with before-and-after examples that are worth studying as pairs rather than reading as advice.
- Few (2006) for dashboard design specifically, and in particular for the argument that a dashboard's defining constraint is the single screen; its catalog of common dashboard failures remains the best diagnostic list available and covers most of what Section 13.1's first draft did wrong.
- Segel and Heer (2010) together with Cairo (2016) for the analytical account of narrative visualization and the broader treatment of visualization as a truth-seeking practice. Read Segel and Heer for the author-driven to reader-driven spectrum that Section 13.3 uses to explain why the go/no-go artifact is a data story wearing a dashboard's clothes; read Cairo on how much a graphic should show, which is the companion problem to Section 13.15's question of what may be left out.
- Healey and Enns (2012) for the survey of the perception research behind Section 13.7, which explains why emphasis is a scarce budget rather than a technique, where the conjunction penalty comes from, and under what conditions the preattentive effect is weaker than the popular summaries suggest.
- Krug (2014) for usability testing as a practice a non-specialist can actually run, including the discipline of not helping the reader, which is the hardest part of Lab 13.2 and the part the book is most useful on.
- Tableau (2026a–2026k) for the authoritative current build paths behind every instruction in Section 13.12 — reference lines and bands, dashboard actions, filter scope, accessibility and alt text, dashboard sizing and device layouts, dynamic titles, multiple-measure axes, mark properties, mark order, Prep multiple-row calculations, and the forecasting options this chapter deliberately declines to use. Where a printed source and the vendor's documentation disagree about an interface, the documentation governs, and it is the reference an analyst should learn to check first.

13.20 References

Bezos, J. (2018). *2017 letter to shareholders*. Amazon.com, Inc. https://www.aboutamazon.com/news/company-news/2017-letter-to-shareholders

Cairo, A. (2016). *The truthful art: Data, charts, and maps for communication*. New Riders.

Correll, M., & Gleicher, M. (2014). Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE Transactions on Visualization and Computer Graphics, 20*(12), 2142–2151. https://doi.org/10.1109/TVCG.2014.2346298

Doncaster, P. (2014). *The UX five-second rules: Guidelines for user experience design's simplest testing technique*. Morgan Kaufmann.

Few, S. (2006). *Information dashboard design: The effective visual communication of data*. O'Reilly Media.

Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten* (2nd ed.). Analytics Press.

Healey, C. G., & Enns, J. T. (2012). Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics, 18*(7), 1170–1188. https://doi.org/10.1109/TVCG.2011.127

Heer, J., & Shneiderman, B. (2012). Interactive dynamics for visual analysis. *Communications of the ACM, 55*(4), 45–54. https://doi.org/10.1145/2133806.2133821

Hullman, J. (2020). Why authors don't visualize uncertainty. *IEEE Transactions on Visualization and Computer Graphics, 26*(1), 130–139. https://doi.org/10.1109/TVCG.2019.2934287

Hullman, J., & Diakopoulos, N. (2011). Visualization rhetoric: Framing effects in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 2231–2240. https://doi.org/10.1109/TVCG.2011.255

Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. Wiley.

Kosslyn, S. M. (2006). *Graph design for the eye and mind*. Oxford University Press.

Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability* (3rd ed.). New Riders.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81–97. https://doi.org/10.1037/h0043158

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 206–213). Association for Computing Machinery. https://doi.org/10.1145/169059.169166

Segel, E., & Heer, J. (2010). Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics, 16*(6), 1139–1148. https://doi.org/10.1109/TVCG.2010.179

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages* (pp. 336–343). IEEE. https://doi.org/10.1109/VL.1996.545307

Spiegelhalter, D., Pearson, M., & Short, I. (2011). Visualizing uncertainty about the future. *Science, 333*(6048), 1393–1400. https://doi.org/10.1126/science.1191181

Tableau. (2026a). *Reference lines, bands, distributions, and boxes*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/reference_lines.htm

Tableau. (2026b). *Actions*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/actions.htm

Tableau. (2026c). *Filter data from your views*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/filtering.htm

Tableau. (2026d). *Author views for accessibility*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/accessibility_create_view.htm

Tableau. (2026e). *Size and lay out your dashboard*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm

Tableau. (2026f). *Format individual parts of the view*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/formatting_specific_titlecaption.htm

Tableau. (2026g). *Forecast options dialog box*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/forecast_options.htm

Tableau. (2026h). *Build a view to compare multiple measures*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/multiple_measures.htm

Tableau. (2026i). *Control the appearance of marks in the view*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/viewparts_marks_markproperties.htm

Tableau. (2026j). *Move marks to the front or back*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/pro/desktop/en-us/move_marks.htm

Tableau. (2026k). *Create multiple-row calculations in Tableau Prep*. Tableau Help. Retrieved July 31, 2026, from https://help.tableau.com/current/prep/en-us/prep_multirow_calculations.htm

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12*(1), 97–136. https://doi.org/10.1016/0010-0285(80)90005-5

Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Graphics Press.

Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science, 211*(4481), 453–458. https://doi.org/10.1126/science.7455683

Appendix A

Python and Google Colab Quick Reference

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · July 2026
Except where otherwise noted, this appendix is licensed under CC BY 4.0.

Appendix Information

PURPOSE

This appendix is a task-indexed reference for the Google Colab and Python actions that recur across the guide. It exists so that a student who already knows what the analysis requires does not have to search four chapters to remember how to perform the step. Every entry states the task, supplies a short runnable pattern, names the result that pattern should produce, and gives one check that confirms it did.

USE THIS APPENDIX WHEN

Consult this appendix when you know the analytic step you need and want the syntax for it: opening a notebook, loading a course file, inspecting a table, filtering, deriving a column, grouping, joining, reshaping, drawing a quick chart, fitting a chapter-specified model, verifying a transformation, diagnosing an error message, or preparing work for submission.

THIS APPENDIX DOES NOT

This appendix is a quick reference. It does not teach Python, does not teach statistical or analytical methods, does not reproduce the chapter labs, and does not supply answers to graded exercises. It catalogs only the commands the guide actually uses, not the full capability of pandas, Matplotlib, statsmodels, or scikit-learn. When an entry raises a question of judgment rather than syntax, it points to the chapter that owns the decision and stops there.

VERSION AND DATE

Version 1.0 · July 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Python and Google Colab quick reference. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Appendix A, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Except where otherwise noted, this appendix is licensed under a Creative Commons Attribution 4.0 International License.
Copyright © 2026 by Jose Mendoza.

Google Colab and Google Drive are products of Google LLC. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Matplotlib, NumPy, and statsmodels are NumFOCUS fiscally sponsored projects. scikit-learn is a NumFOCUS-affiliated project. ChatGPT, Claude, Gemini, GitHub Copilot, and Tableau are trademarks of their respective owners. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this appendix, including outlining, preliminary drafts, revision of prose, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the appendix's scope, boundaries, and content, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and every code pattern was executed and its stated result confirmed. Responsibility for the accuracy, originality, and final form of this appendix rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

COMPANION FILES

A runnable Appendix_A_Quick_Reference.ipynb built to pass Restart and run all, the small practice files and the larger synthetic demonstration files this appendix uses, a standalone command index, and the maintained package-version record are in the Applied Business Analytics companion repository.

How to Read an Entry

Entries in this appendix are organized by task, not by function name. You should be able to find a command by describing what you are trying to do, which is why Table A.1 and the command index in Section A.16 both begin with the task rather than the syntax.

Every entry has the same four parts. The code caption states the task. The code block supplies a short pattern. The italic line that follows names the result the pattern should produce — usually a shape or a type rather than a number, because your file is not the one this appendix was written against. The final line supplies one verification: a check that would fail if the step did something other than what you intended.

That fourth part is not decoration. The guide's second discipline is predict-then-verify (Section 1.7), and a reference that supplies syntax without a check trains the wrong habit. Where an entry transforms data, the verification is mandatory, not optional.

CONCEPT

Copy, Adapt, Verify

A pattern in this appendix is a starting point, not an answer. Copy it, replace every placeholder with the field names your file actually carries, run it, and then run the verification line. Code that executes without an error has proved only that it is valid Python. It has not proved that it computed what you meant.

The same rule governs code an AI assistant drafts for you. Appendix C supplies prompt templates for asking an assistant to produce any pattern in this appendix; Appendix D is where the exchange, the correction, and the verification are recorded.

Source: Course concept developed for this guide.

Placeholders appear in square brackets and capital letters. [FILE] stands for a file name, [COLUMN] for a column in your own table, [KEY] for a join key, and [WINDOW] for a date range you have declared. Replace all of them before running. A pattern that still contains a bracketed placeholder has not been adapted.

Most examples run against a small practice file, practice_transactions.csv, which ships with this appendix: twenty order lines, twelve orders, eight customers, and $1,312.00 of revenue, shaped exactly like StyleCraft's transactions table and small enough to check with a calculator. It is a practice file, not an assignment file. No number in this appendix answers a graded exercise. Sections A.11 and A.12 need more rows than twelve orders to be worth looking at, so the companion repository also ships synthetic demonstration files at scale; they are clearly labeled, they are not StyleCraft data, and no number in them means anything. Against your own file, expect your output to differ.

Table A.1

What to consult, by what you are trying to do

If you need to…	Go to
Open, run, save, or recover a notebook	A.1
Load a course file, or write a processed file back out	A.2
Remember a piece of Python syntax, or read an error message	A.3
Start a notebook with the right imports and an environment record	A.4

If you need to…	Go to
See what is in a table you have just loaded	A.5
Choose rows or columns, sort, or build a derived variable	A.6
Handle missing values, duplicates, wrong types, or dates	A.7
Summarize by group, or change the grain of a table	A.8
Combine two tables without corrupting either	A.9
Move between wide and long, or between transaction and customer grain	A.10
Draw a histogram, bar, line, scatter, or box plot	A.11
Split, fit, predict, score, or cross-validate	A.12
Prove that a step did what you intended	A.13
Diagnose an error or an implausible result	A.14
Export figures and tables and submit the work	A.15
Find a command when you cannot remember its name	A.16

Table A.2 records the StyleCraft tables this appendix loads and the grain of each, because almost every mistake in the sections that follow is a grain mistake wearing a different costume. One row of transactions is one order line, not one order and not one customer; a metric computed at the wrong grain is wrong even when every command in the cell is correct.

Table A.2

The StyleCraft tables this appendix refers to, and the grain of each

Table	One row is	Key
`transactions`	one order line	`order_line_id`
`customers`	one customer	`customer_id`
`products`	one product (SKU)	`product_id`
`stores`	one physical store	`store_id`
`campaigns`	one marketing campaign	`campaign_id`
`customer_features`	one customer, at a stated snapshot	`customer_id`
`marketing_daily`	one calendar day	`date`
`ab_test`	one randomized customer	`customer_id`
`transactions_enriched`	one order line, with dimensions joined on	`order_line_id`

A.1 Getting Started in Google Colab

Google Colab runs a Python notebook on a Google server rather than on your machine. Nothing needs to be installed. Two consequences follow from that, and both cause more lost work than any syntax error in this appendix: the machine you are given is temporary, and the variables you create live in that machine's memory rather than in the notebook file.

In other words, the notebook is saved; the runtime is not. Files you upload and variables you define disappear when the runtime is recycled, which happens after a period of inactivity and whenever you choose Restart. Everything in Section A.15 about running a notebook from top to bottom before submission follows from this one fact.

A.1.1 The First Five Minutes

Work through this list at the start of every session.

98. Open the notebook from the companion repository, from Brightspace, or from your own Google Drive. Opening from the repository gives you a read-only view; use File → Save a copy in Drive before you edit.
99. Confirm the notebook is connected. The Connect button at the top right becomes a RAM and Disk indicator once a runtime is attached.
100. Rename the file so it identifies you and the assignment, for example Assignment1_Mendoza.ipynb. Do not submit a file still called Copy of ....
101. Run the first cell and read its output. Code A.1 is that cell.
102. Load the data you need before you write any analysis, and check its shape (Section A.5) before you compute anything from it.

Code A.1. Confirm where you are and what you have

```
import sys
from pathlib import Path

print("Python:", sys.version.split()[0])
print("Working directory:", Path.cwd())

here = sorted(p.name for p in Path(".").iterdir() if p.is_file())
print(f"{len(here)} file(s) here:", here)
```

Expected result: the Python version, the working directory (usually /content in Colab), and a complete count and listing of the files the runtime can see.

Verify: the listing is complete rather than truncated, so a file absent from it really is absent — a truncated listing cannot support that conclusion. If the file you need is missing, Section A.2 shows how to put it there.

A.1.2 Cells, Running, and Stopping

A notebook is a sequence of cells. Code cells contain Python and produce output below themselves; text cells contain Markdown and carry your explanations. A cell's output shows only its last expression, which is why cells in this guide print results explicitly rather than relying on that behavior. Table A.3 lists the notebook actions you will use and what each one does to the runtime.

Table A.3

Notebook actions and what each one does

To do this	Use this	What happens
Run one cell	Shift+Enter, or the play button	Runs the cell and moves to the next
Run one cell and stay	Ctrl+Enter	Runs the cell in place
Run everything	Runtime → Run all	Runs every cell from the top in order
Stop a running cell	The stop button, or Runtime → Interrupt	Ends the cell; variables already created survive
Clear the memory	Runtime → Restart session	Every variable and imported module is lost; the notebook text is not
Start completely clean	Runtime → Restart and run all	The only honest test that a notebook reproduces
Clear the printed output	Edit → Clear all outputs	Removes results, keeps the code

Restarting matters more than it appears to. A notebook that runs correctly in the order you happened to execute the cells may fail completely when read from top to bottom, because a variable you defined in a cell you later

deleted is still sitting in memory. Restart and run all is the check that catches this, and Section A.15 requires it before submission.

A.1.3 Saving and Downloading

Colab autosaves to Google Drive, but only after you have saved a copy there. Use File → Save a copy in Drive on any notebook you opened from a link. To hand work in, use File → Download and choose .ipynb, which preserves code, output, and text together. Download .py only when the course asks for a script, and produce a PDF only when the assignment asks for one; a PDF is a picture of the work, not the work.

A.2 Files and Paths

The guide uses one file workflow by default: read the course file straight from the companion location into a DataFrame, in one line, with the date columns parsed as dates. Uploading and mounting Drive are alternatives worth knowing, but they are not the default, because an uploaded file has to be uploaded again after every runtime restart, and a mounted Drive makes the notebook depend on one person's folder structure.

Code A.2. Read a course file from the companion location

```
import pandas as pd

# One line to change if the course files move. The current value is
# published on the companion resource page; do not retype the full
# address in every cell.
DATA = ("https://raw.githubusercontent.com/jrmst102/busin"
        "essanalytics/main/data/")

transactions = pd.read_csv(DATA + "practice_transactions.csv",
                           parse_dates=["order_date"])
print(transactions.shape)

# An Excel workbook is read the same way, naming the sheet.
# book = pd.read_excel(DATA + "[FILE].xlsx", sheet_name="[SHEET]")
```

Expected result: a tuple of (rows, columns). For the practice file it is (20, 13).

Verify: the row count matches what the file is documented to contain, and order_date is a date rather than text — confirm with transactions.dtypes (Section A.5).

Two arguments in that call are doing real work. The parse_dates list converts the named columns to true dates at read time, which is cheaper and safer than converting later; a date left as text sorts alphabetically and silently produces a wrong answer. And assigning the location to DATA once means a moved file is a one-line repair rather than a search through the notebook.

Code A.3. Upload a file from your own computer (Colab only)

```
from google.colab import files

uploaded = files.upload()          # opens a file chooser
print(list(uploaded.keys()))

import pandas as pd
transactions = pd.read_csv("[FILE].csv", parse_dates=["order_date"])
```

Expected result: a file chooser, then the name of each uploaded file.

Verify: the file appears in the listing produced by Code A.5. Remember that an uploaded file is lost on restart and has to be uploaded again.

Code A.4. Connect Google Drive (Colab only)

```
from google.colab import drive
drive.mount("/content/drive")

import pandas as pd
path = "/content/drive/MyDrive/[FOLDER]/[FILE].csv"
transactions = pd.read_csv(path, parse_dates=["order_date"])
```

Expected result: an authorization prompt, then the message Mounted at /content/drive.

Verify: the path exists before you read it. Use Code A.5 on the folder rather than guessing at the spelling of MyDrive.

Mount Drive when an assignment genuinely requires files that live in your own Drive. Do not mount it in a notebook you intend to submit for a group, because the path will not resolve on anyone else's account.

Code A.5. See what the runtime can actually find

```
from pathlib import Path

for p in sorted(Path(".").glob("*.csv")):
    print(p.name, f"{p.stat().st_size:,} bytes")
```

Expected result: one line per CSV file in the working directory, with its size. No output at all means there are no CSV files there.

Verify: the file you are about to load is on that list, spelled exactly as you will type it. File names are case-sensitive.

Code A.6. Write a processed file back out, and confirm it landed

```
orders = (transactions
          .groupby("order_id", as_index=False)["line_revenue"]
          .sum())

orders.to_csv("orders_summary.csv", index=False)

check = pd.read_csv("orders_summary.csv")
print("rows written:", len(check))
```

Expected result: the number of rows in the file you just wrote. For the practice file it is 12, one per order.

Verify: read the file back and compare its row count and its total to the frame you wrote. index=False keeps pandas from adding an unnamed index column that will confuse the next person who opens the file.

A file written into the working directory is on the temporary machine, not on yours. Download it with File → Download, or write it into a mounted Drive folder, before the runtime is recycled.

A.3 The Small Amount of Python You Need Repeatedly

This section is a memory aid for the handful of language features the guide's notebooks actually use. It is not an introduction to Python, and it deliberately omits classes, comprehensions beyond the simplest form, and everything else that a marketing analyst can be productive without.

Code A.7. Assignment, the basic types you will meet, and comments

```
store_id = "S09"                        # a string
unit_price = 88.00                      # a float (a decimal number)
quantity = 2                            # an integer (a whole number)
is_occasionwear = True                  # a Boolean: True or False
line_revenue = quantity * unit_price    # 176.0
```

```
metros = ["NYC Urban", "NYC Suburban", "Resort"]   # a list, ordered
thresholds = {"caution": 0.05, "stop": 0.10}       # a dictionary

print(line_revenue, metros[1], thresholds["stop"])
```

Expected result: 176.0 NYC Suburban 0.1.

Verify: list positions start at 0, so metros[1] is the second element. A dictionary is read by key, not by position.

The single equals sign assigns; the double equals sign compares. Anything after a # on a line is a comment, ignored by Python and read by the next person, who is usually you three weeks later. Indentation is not cosmetic: the four spaces at the start of a line are what tell Python which lines belong inside an if statement or a function.

Code A.8. Function calls, keyword arguments, comparisons, and if

```
n = len(transactions)                    # positional argument
total = round(transactions["line_revenue"].sum(), 2)

# A keyword argument is named at the call site. Both lines below
# do the same thing; the second says what the 2 means.
head_rows = transactions.head(3)
head_rows = transactions.head(n=3)

big = transactions["line_revenue"] > 100       # one answer per row
both = ((transactions["quantity"] >= 2)        # use & and |, not
        & (transactions["discount_pct"] == 0)) # "and" and "or"

if total > 1000 and n == 20:
    print(f"{n} lines totalling ${total:,.2f}")
else:
    print("this is not the file the cell expects")
```

Expected result: for the practice file, 20 lines totalling $1,312.00.

Verify: a keyword argument is matched by name, so its position does not matter and its meaning is visible. Named arguments recur throughout this appendix — parse_dates=, dropna=, validate=, random_state= — and reading them is how you know what a cell is doing. A comparison against a column returns a column of True and False, one per row, not a single answer. Wrap each condition in parentheses and join them with & or |; Python's and and or do not work on columns.

The f before the quotation mark makes an f-string, which substitutes the value of anything in braces. The :,.2f inside the braces asks for thousands separators and two decimal places. Use it whenever you print money.

Code A.9. Write a rule once, in a function, and reuse it

```
def contribution(revenue, cost):
    """Contribution dollars for one line. Both arguments in dollars."""
    return revenue - cost

transactions["contribution"] = contribution(
    transactions["line_revenue"], transactions["line_cost"])

print(round(transactions["contribution"].sum(), 2))
```

Expected result: one number: total contribution dollars across the file.

Verify: the sum of the new column equals total revenue minus total cost. A rule written once in a function cannot drift between two cells, which is the whole argument for putting it there.

A.3.1 Reading an Error Message

Python error messages are read from the bottom upward. The last line names the error type and gives the specific complaint; that is the line to read first. The lines above it are the traceback, showing where the failure happened, and the topmost lines are usually inside the library rather than in your code.

Code A.10. Provoke an error deliberately and read what it says

```
try:
    transactions["revenue"].sum()   # the column is line_revenue
except KeyError as exc:
    print("KeyError:", exc)
    print("columns that do exist:", list(transactions.columns))
```

Expected result: KeyError: 'revenue', followed by the list of column names the table actually carries.

Verify: the printed list is the answer. A KeyError is almost always a spelling, a case difference, or a column that a previous cell renamed or dropped. Section A.14 catalogs the errors you will meet most.

A.4 Core Imports and the Reproducibility Block

Every notebook in this guide opens with the same three imports. Chapter-specific libraries are imported in the chapter that needs them, at the point of need, so that a reader can see which section introduced a dependency.

Code A.11. The canonical import block

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Expected result: no output. An import that succeeds is silent.

Verify: if any line raises ModuleNotFoundError, see Section A.14.5. The aliases np, pd, and plt are conventions, not requirements, but every example in this guide and in the companion notebooks assumes them.

Table A.4 names the libraries the guide depends on, what each is used for, and the chapter that introduces it. pandas supplies the DataFrame (McKinney, 2010), NumPy the numeric array beneath it (Harris et al., 2020), and Matplotlib the charts (Hunter, 2007); scikit-learn (Pedregosa et al., 2011) and statsmodels (Seabold & Perktold, 2010) arrive later, in Part II.

Table A.4

The libraries the guide uses, and the chapter that introduces each

Library	Used for	Introduced in
`pandas`	Tables: loading, cleaning, grouping, joining	Chapter 4
`numpy`	Numeric arrays and the isclose and allclose checks	Chapter 4
`matplotlib.pyplot`	Charts drawn in the notebook	Chapter 4
`scikit-learn`	Scaling, clustering, splits, pipelines, models, metrics	Chapters 6 and 8
`statsmodels`	Regression with interpretable coefficients; smoothing; proportion tests	Chapters 7, 10, and 11
`IPython.display`	Showing more than one result from a single cell	Chapter 4

Do not add imports the notebook does not use. An unused import is a small thing, but it is also a signal that code arrived from somewhere else without being read — which is exactly the habit this guide is built to prevent.

Code A.12. The environment record

```
import sys
import matplotlib
import sklearn
import statsmodels

print("Python      :", sys.version.split()[0])
print("pandas      :", pd.__version__)
print("numpy       :", np.__version__)
print("matplotlib  :", matplotlib.__version__)
print("scikit-learn:", sklearn.__version__)
print("statsmodels :", statsmodels.__version__)
```

Expected result: six version numbers, one for Python and one for each library the guide depends on.

Verify: compare these against the version record in the companion repository when a result you expect to reproduce does not. Place this cell near the top of any notebook you submit; it is the cheapest possible insurance against an unreproducible result.

Version numbers are not printed in this appendix's prose on purpose. Libraries change between terms, and a number frozen into a book ages badly. The maintained environment record in the companion repository is the authoritative statement of which versions the course notebooks were run against.

A.5 Inspecting a DataFrame

Run these five commands on every file, in this order, before computing anything from it. They take under a minute and they catch most of the failures that would otherwise appear much later, disguised as an analytical result.

Code A.13. Shape, column names, and the first rows

```
print("rows, columns:", transactions.shape)
print(list(transactions.columns))
print(transactions.head())
```

Expected result: a (rows, columns) tuple, the column names as a list, and the first five rows.

Verify: the row count matches what the file is documented to contain. A shape of (1, 13) usually means the delimiter was wrong; a shape with far more columns than expected usually means a stray comma in the source.

Code A.14. Types, structure, and summaries

```
from IPython.display import display

display(transactions.dtypes)
transactions.info()
display(transactions.select_dtypes(include="number").describe().round(2))
display(transactions.select_dtypes(exclude="number").describe())
```

Expected result: the type of each column; a structural summary with non-null counts; a numeric summary of count, mean, standard deviation, minimum, quartiles, and maximum; and, for the non-numeric columns, a count of distinct values with the most frequent one.

Verify: every column that should be a number is a number and every column that should be a date is a date. A price column reported as anything other than a numeric type — object or str, depending on the pandas version — is text, usually carrying currency symbols; see Code A.26.

A notebook renders only the last expression in a cell, so a cell listing four results shows one of them. The display function from IPython prints each one explicitly. Reading describe on the numeric columns is where implausible values first announce themselves: a maximum quantity of 999, a minimum price of zero, a mean far from the median.

Code A.15. Missing values, distinct values, and a frequency table

```
print(transactions.isna().sum())
print(transactions.nunique())
print(transactions["channel"].value_counts(dropna=False))
print(transactions["channel"].value_counts(normalize=True,
                                           dropna=False).round(3))
```

Expected result: a count of missing values per column, a count of distinct values per column, then the counts and the shares for one categorical column.

Verify: dropna=False is not optional. Without it, missing values vanish from the frequency table and the shares are computed on a denominator you did not choose. A category list longer than you expected usually means inconsistent labels; see Code A.21.

Table A.5 records what each of those commands is actually for. Read it as a set of questions rather than a set of functions: the question is what you have when you open an unfamiliar file.

Table A.5

What each inspection command answers

Command	The question it answers
`.shape`	How many rows and columns did I actually get?
`.columns`	What are the fields called, exactly?
`.head() / .tail()`	What does a row look like, and does the end of the file look like the start?
`.dtypes`	Is each column stored as the type it should be?
`.info()`	Which columns have missing values, and how large is the table in memory?
`.describe()`	Are the numeric ranges plausible? Is the mean far from the median?
`.isna().sum()`	Where is data missing, and how much?
`.nunique()`	Which columns are keys, and which are categories?
`.value_counts()`	What labels exist in this category, and how common is each?

Code A.16. Row count and distinct-key check

```
print("rows                 :", len(transactions))
print("distinct order lines :", transactions["order_line_id"].nunique())
print("distinct orders      :", transactions["order_id"].nunique())
print("distinct customers   :", transactions["customer_id"].nunique())

assert transactions["order_line_id"].is_unique, "order_line_id is not a key"
```

Expected result: four counts, and silence from the assertion. For the practice file: 20 rows, 20 order lines, 12 orders, 8 customers.

Verify: the count of distinct order lines equals the row count. If it does not, the table has duplicates (Code A.25) and every total computed from it will be too high.

That last check is the one to run first on any fact table. It establishes the grain: what one row of this table represents. Chapter 3 owns that concept (Section 3.3); this appendix only supplies the command that confirms it.

A.6 Selecting, Filtering, Sorting, and Creating Columns

These six patterns account for most of the data handling in the guide's labs. Each of them produces a new object rather than changing the one you started from, which is deliberate: the original file stays available as evidence, and every step is reversible by rerunning the notebook from the top.

Code A.17. Select one column and several columns

```
revenue = transactions["line_revenue"]                    # a Series
keys = transactions[["order_id", "customer_id", "line_revenue"]]  # a frame

print(type(revenue).__name__, type(keys).__name__, keys.shape)
```

Expected result: Series DataFrame (20, 3).

Verify: single brackets return one column as a Series; double brackets return a table, even when the list has one name in it. Many puzzling errors later in a notebook come from having one where you meant the other.

Code A.18. Filter on one condition, several conditions, and a list

```
store_lines = transactions[transactions["channel"] == "Store"]

full_price_store = transactions[(transactions["channel"] == "Store")
                                & (transactions["discount_pct"] == 0)]

digital = transactions[transactions["channel"].isin(["Online", "App"])]

print(len(store_lines), len(full_price_store), len(digital))
```

Expected result: three row counts. Each is a subset, so each is no larger than the original.

Verify: the counts of mutually exclusive, exhaustive subsets add back to the original row count. If they do not, a category label is missing from your isin list, or a row has a missing value in the filtered column — which is excluded by every comparison, including inequalities.

Code A.19. Sort, with a tie-break that makes the order reproducible

```
# Sorted on the measure alone: ties fall wherever the file put them.
top = transactions.sort_values("line_revenue", ascending=False)

# Sorted with an identifier as the tie-break: reproducible anywhere.
stable = transactions.sort_values(["line_revenue", "order_line_id"],
                                  ascending=[False, True])

print(stable[["order_line_id", "line_revenue"]].head(3))
```

Expected result: the three largest lines from the reproducibly sorted frame, largest first.

Verify: print stable rather than top. Sorting on the measure alone leaves ties in whatever order the file happened to arrive in, so a ranked list can differ between machines. Adding the identifier as a second sort key makes the ordering reproducible by anyone.

Code A.20. Create a derived variable from an explicit rule

```
transactions["gross_line_revenue"] = (transactions["quantity"]
                                      * transactions["unit_price"])
transactions["discount_dollars"] = (transactions["gross_line_revenue"]
                                    - transactions["line_revenue"])
transactions["is_discounted"] = transactions["discount_pct"] > 0

print(transactions.loc[transactions["is_discounted"],
                       ["order_line_id", "gross_line_revenue",
                        "discount_dollars", "line_revenue"]])
```

Expected result: one row per discounted line, showing the arithmetic.

Verify: gross revenue minus discount dollars equals line revenue on every row. Check it with np.allclose rather than by eye: np.allclose(transactions["gross_line_revenue"] - transactions["discount_dollars"], transactions["line_revenue"]).

A derived variable is a definition, and a definition belongs in writing. When the derived column is a named metric — average order value, repeat purchase rate, contribution margin — the numerator, denominator, window, filters, and grain are cataloged in Appendix B, and the code here should implement that entry rather than inventing a second version of it.

Code A.21. Rename a column and recode a category

```
# Rename onto a NEW name. Rebinding transactions here would break
# every later cell that still refers to line_revenue.
renamed = transactions.rename(columns={"line_revenue": "revenue"})
print(list(renamed.columns)[-3:])

channel_map = {"Online": "Digital", "App": "Digital", "Store": "Store"}
transactions["channel_group"] = transactions["channel"].map(channel_map)

print(transactions["channel_group"].value_counts(dropna=False))
assert transactions["channel_group"].notna().all(), "an unmapped label"
```

Expected result: the last three column names of the renamed copy, then the counts for each new group, and silence from the assertion.

Verify: map turns any label missing from the dictionary into a missing value, silently. The assertion is what turns that silence into a message. List the distinct labels with value_counts before writing the dictionary, not after.

Code A.22. Assign to a copy, never to a slice of a slice

```
store = transactions[transactions["channel"] == "Store"].copy()
store["revenue_per_unit"] = store["line_revenue"] / store["quantity"]

print(store["revenue_per_unit"].round(2).head())
```

Expected result: one revenue-per-unit figure per store line.

Verify: the new column exists on store and not on transactions. Without .copy(), pandas may warn about setting a value on a copy of a slice, and in recent versions the assignment simply does not reach the original table. Filter, copy, then assign — in that order.

A.7 Missing Values, Duplicates, Types, and Dates

This section supplies the commands. It does not supply the decisions. Whether a missing discount means no discount or an unrecorded one, whether an outlier is an error or a legitimate extreme, and whether a near-duplicate is a re-export or a genuine second order are all analytic questions that Chapter 4 owns and that a command cannot answer.

> **CONCEPT**
>
> **A Command Is Not a Justification**
>
> Every entry in this section changes the data. Running one of them records nothing about why it was appropriate. The remedy you choose, the rows it touched, and the bias it introduces belong in the verification log Chapter 4 requires, and — when an AI assistant proposed the remedy — in the record Appendix D requires.
>
> The practical test is simple. If someone asked why the file now has forty fewer rows than the file you were given, could you answer without rerunning anything?

> Source: Course concept developed for this guide.

Code A.23. Find the missing values and size the problem

```
missing = (transactions.isna().sum().rename("missing").to_frame()
           .assign(share=lambda d: (d["missing"] / len(transactions))
                   .round(4)))

print(missing[missing["missing"] > 0])
```

Expected result: one row per column that has any missing values, with the count and the share of the file. No output means nothing is missing.

Verify: compare the columns that appear against the data dictionary. A column documented as always present but missing in 3% of rows is a data problem to trace, not a value to impute.

Code A.24. Fill or drop, only after the decision has been made

```
before = len(transactions)

# Decision recorded in the verification log: an absent discount on a
# full-price line means no discount was applied.
transactions["discount_pct"] = transactions["discount_pct"].fillna(0.0)

# Decision: a line with no quantity cannot be valued, so it is excluded
# from revenue work and counted in the log.
computable = transactions.dropna(subset=["quantity"])

print("rows:", before, "| rows with a usable quantity:", len(computable))
```

Expected result: two counts. The difference between them is the number of rows the decision removed.

Verify: print the row count before and after, every time, and record both. A fill that changes a total and a drop that changes a row count are the two most common sources of a number that no one can reconcile later.

Code A.25. Exact duplicates and duplicates on the identity columns

```
exact = int(transactions.duplicated().sum())

identity = ["order_id", "customer_id", "product_id",
            "quantity", "unit_price"]
near = int(transactions.duplicated(subset=identity).sum())

print("exact duplicates:", exact,
      "| duplicates on the identity columns:", near)

deduped = transactions.drop_duplicates().copy()
```

Expected result: two counts. The second is at least as large as the first.

Verify: row count before minus exact duplicates equals row count after. A near-duplicate — the same order re-exported with a different timestamp — is invisible to the first check and visible to the second, which is why both are run.

Code A.26. Coerce text to a number, and a count to a nullable integer

```
# Run this against the raw file, where the defects actually live.
raw = pd.read_csv(DATA + "[RAW FILE].csv")
before_na = int(raw["unit_price"].isna().sum())
```

```
price_text = raw["unit_price"].astype("string")
raw["unit_price"] = pd.to_numeric(
    price_text.str.replace(r"[$,]", "", regex=True), errors="coerce")

after_na = int(raw["unit_price"].isna().sum())

# A count that can legitimately be missing needs a nullable
# integer type; plain int cannot hold one.
raw["quantity"] = raw["quantity"].astype("Int64")

print(before_na, "->", after_na,
      "| quantity dtype:", raw["quantity"].dtype)
```

Expected result: two missing-value counts, and Int64 with a capital I as the new type.

Verify: the two counts must be equal. errors="coerce" turns anything it cannot parse into a missing value rather than raising, so a rise in the count is the coercion telling you it failed on rows you have not looked at. Find them before continuing.

Table A.6 pairs the type symptoms you will actually see with the coercion that repairs each. It lists commands, not verdicts: whether a given value in a given file should be repaired, excluded, or traced back to its source is Chapter 4's question, and the defect catalog that governs the StyleCraft raw file is Chapter 4's too.

Table A.6

Type problems and the coercion that repairs each

Symptom	Cause	Repair
A price column is not a numeric dtype	Currency symbols or thousands separators in the text	Strip with .str.replace, then pd.to_numeric
Dates sort alphabetically	The column is still text	pd.to_datetime with errors="coerce"
A count column will not accept a missing value	Plain int cannot hold one	Cast to the nullable Int64
A 0/1 flag behaves like text	It was read as "True"/"False" strings	Map explicitly, then .astype(int)
A category has more labels than expected	Case and spelling variants	Standardize with .str.strip().str.title(), then .map

Code A.27. Parse dates with explicit error handling, then read the calendar

```
transactions["order_date"] = pd.to_datetime(
    transactions["order_date"], errors="coerce")
unparsed = int(transactions["order_date"].isna().sum())

transactions["order_year"] = transactions["order_date"].dt.year
transactions["order_month"] = transactions["order_date"].dt.to_period("M")
transactions["order_weekday"] = transactions["order_date"].dt.day_name()

print("dates that would not parse:", unparsed)
assert unparsed == 0, "investigate every unparsed date before continuing"
```

Expected result: a count of unparsed dates, and silence from the assertion when that count is zero.

Verify: unparsed dates are not a nuisance to be dropped. A date that fails to parse is usually a different format in part of the file, which means part of the file came from somewhere else.

Code A.28. Check implausible values with Boolean conditions

```
off_catalog = transactions[(transactions["unit_price"] < 24)
                           | (transactions["unit_price"] > 90)]
```

```
big_baskets = transactions[transactions["quantity"] > 20]
negative = transactions[transactions["line_revenue"] < 0]

print("lines priced outside the catalog band:", len(off_catalog))
print("lines with an implausible quantity   :", len(big_baskets))
print("lines with negative revenue          :", len(negative))
```

Expected result: three counts, each ideally zero. Any non-zero count is a list of rows to inspect, not a list of rows to delete.

Verify: the bounds come from the business, not from the data. StyleCraft's catalog runs from $24 to $90, so a $4.99 line is a defect even though it is a perfectly valid number. Write the bound you expect before you look at what the file contains.

A.8 Grouping and Aggregation

Grouping changes the grain of a table, and a change of grain changes what a row means. Lead with the grain, not with the syntax: state what one input row represents, what one output row should represent, and which total must survive the change. Then write the groupby.

> **CONCEPT**
>
> **Every Aggregation Has Two Grains and One Invariant**
>
> Before running any groupby, write four things down: the input grain, the output grain, the aggregation rule for each column, and the total or count that must be identical on both sides.
>
> Rolling order lines up to orders should preserve total revenue exactly. Rolling orders up to customers should preserve both total revenue and the count of orders. If an invariant you named does not hold, the aggregation is wrong — however sensible the output table looks.
>
> Source: Course concept developed for this guide.

Two arguments appear on every groupby in this guide. Pass dropna=False so that rows with a missing group key are visible rather than discarded, and pass observed=True whenever a grouping key is or could become a categorical, so that combinations which never occurred do not appear as empty rows and quietly change a denominator. Treat both as house rules rather than as options.

Code A.29. Group by one key, with named aggregations

```
# Input grain: one order line.  Output grain: one channel.
# Rule: lines counted, orders counted distinctly, units and
# revenue summed, revenue median reported alongside the mean.
# Invariant: summed revenue equals the file's total revenue.
by_channel = (transactions
              .groupby("channel", as_index=False,
                       dropna=False, observed=True)
              .agg(lines=("order_line_id", "size"),
                   orders=("order_id", "nunique"),
                   units=("quantity", "sum"),
                   revenue=("line_revenue", "sum"),
                   median_line=("line_revenue", "median")))

print(by_channel)
```

Expected result: one row per channel, with five columns you named yourself.

Verify: the revenue column sums to the file's total revenue. Named aggregation — new_name=("column", "function") — is the house style because it produces readable column names instead of a multi-level index nobody can index into.

The two keyword arguments are deliberate. as_index=False returns a flat table rather than one indexed by the grouping key, which is what you want when the result feeds a merge or a chart. And dropna=False makes the treatment of missing group keys explicit; without it pandas drops those rows without comment, and the shares you compute in Code A.31 will be shares of a denominator you did not choose.

Code A.30. Group by several keys and return a flat table

```
# Input grain: one order line.  Output grain: one month-channel
# pair.  Invariant: total revenue is unchanged.
transactions["order_month"] = (transactions["order_date"]
                               .dt.to_period("M"))

grid = (transactions
        .groupby(["order_month", "channel"],
                 dropna=False, observed=True)
        .agg(revenue=("line_revenue", "sum"))
        .reset_index())

print(grid.head())
```

Expected result: one row per month-and-channel combination that appears in the data.

Verify: the revenue total is unchanged from the ungrouped file. Use reset_index() when a groupby without as_index=False has left the keys in the index; observed=True keeps categorical keys from generating empty combinations that never occurred.

Code A.31. Counts, rates, and shares computed from pooled totals

```
# Input grain: one order line.  Output grain: one channel.
# revenue_per_order is an ORDER-level measure, so its denominator
# is a distinct order count, never a row count.
by_channel = (transactions
              .groupby("channel", dropna=False, observed=True)
              .agg(orders=("order_id", "nunique"),
                   customers=("customer_id", "nunique"),
                   revenue=("line_revenue", "sum")))

by_channel["revenue_share"] = (by_channel["revenue"]
                               / by_channel["revenue"].sum())
by_channel["revenue_per_order"] = (by_channel["revenue"]
                                   / by_channel["orders"])

print(by_channel.round(3))
```

Expected result: one row per channel with three counts and two computed measures.

Verify: the shares sum to 1. Compute every rate from pooled numerators and denominators, as here. Averaging a rate that is already a rate gives each group equal weight regardless of size, which is a different — and almost always wrong — number.

A distinct count is not a row count. nunique on order_id counts orders; size counts order lines. Choosing the wrong one is the most common way an order-level metric ends up computed at line grain, and the resulting average order value is always too low. Appendix B's entry for average order value states the grain explicitly for exactly this reason.

Code A.32. Change the grain, and prove the change preserved the total

```
# Input grain: one order line.  Output grain: one order.
# Invariant: total revenue, and the count of distinct orders.
orders = (transactions
          .groupby("order_id", as_index=False, dropna=False)
          .agg(customer_id=("customer_id", "first"),
               order_date=("order_date", "min"),
               lines=("order_line_id", "size"),
               units=("quantity", "sum"),
               order_revenue=("line_revenue", "sum")))

assert len(orders) == transactions["order_id"].nunique()
assert np.isclose(orders["order_revenue"].sum(),
                  transactions["line_revenue"].sum())
assert (transactions.groupby("order_id")["customer_id"]
        .nunique().eq(1).all()), "an order carries more than one customer"

print("orders:", len(orders),
      "| AOV:", round(orders["order_revenue"].mean(), 2))
```

Expected result: the order count and the average order value. For the practice file: 12 orders, AOV $109.33.

Verify: three assertions, and each catches a different failure. The first catches a grain that is not what you declared. The second catches revenue lost or duplicated in the roll-up. The third catches the quiet danger in "first": it is honest only when the attribute is constant within the group.

Table A.7 records those four statements for that one example, and is the form to copy into a verification log before every aggregation you write.

Table A.7

The four things to record before every aggregation

Record	Practice-file example
Input grain	One order line
Output grain	One order
Aggregation rule per column	Revenue and units summed; date minimized; customer taken as first
Invariant	Total revenue is $1,312.00 before and after

A.9 Joining Tables

Joins deserve more verification than any other operation in this appendix, because a join failure is both easy to cause and hard to see. A join that quietly duplicates rows raises every total in the notebook, and every downstream number remains internally consistent while being wrong.

CONCEPT

Declare the Cardinality, Then Let the Code Enforce It

Before writing a merge, say aloud what the relationship is: many transaction lines to one store, many orders to one customer, one campaign to one performance row. Then pass that statement to the validate argument so pandas refuses the join if the data disagrees.

Bracket the merge with a row count before and an assertion after. A left join that changes the row count has not enriched the table; it has multiplied it.

Source: Course concept developed for this guide.

Code A.33. The validated merge

```
# Enrich onto a NEW name. Rebinding transactions here would mean
# the next cell that merges onto it collides with these columns.
rows_before = len(transactions)

enriched = transactions.merge(
    stores[["store_id", "store_name", "store_type"]],
    on="store_id",
    how="left",
    validate="many_to_one",
    indicator=True)

assert len(enriched) == rows_before, "the join changed the row count"
print(enriched["_merge"].value_counts())
print("lines with no store row:",
      int(enriched["store_type"].isna().sum()))

enriched = enriched.drop(columns="_merge")
```

Expected result: a breakdown of matched and unmatched rows, then a count of lines that found no store. For the practice file that count is 10 — the lines whose store_id is the reserved value ONLINE.

Verify: the row count is unchanged and the unmatched count is a number you can explain. An unmatched count you cannot explain is a data problem, not a rounding detail. Drop the indicator column once you have read it; a verification artifact does not belong in a certified table. enriched is the line-grain table the rest of this appendix uses when it needs store attributes, and Code A.40 rolls it up for the charts.

Three arguments carry the weight. The on argument names the key explicitly rather than letting pandas guess from matching column names. The validate argument states the expected cardinality and raises a MergeError when the data violates it. And indicator=True adds a temporary _merge column recording, for each row, whether it matched — which is how the unmatched rows become visible instead of becoming missing values.

Code A.34. Inspect the unmatched keys instead of ignoring them

```
unmatched = sorted(set(transactions["store_id"])
                   - set(stores["store_id"]))

print("store_id values with no row in stores:", unmatched)
print("lines affected:",
      int(transactions["store_id"].isin(unmatched).sum()))
```

Expected result: the list of key values present in the left table and absent from the right, and the number of rows they account for.

Verify: every value on that list is either a documented reserved value — ONLINE and APP in StyleCraft's transactions — or a defect. If you cannot classify it into one of those two categories, do not proceed.

Code A.35. Refuse a many-to-many join rather than discover it later

```
# A right-hand table must carry at most one row per key for a
# many-to-one join to be honest. This one carries two.
right = pd.concat([products[["product_id", "unit_price"]]] * 2)

try:
    transactions.merge(right, on="product_id", how="left",
                       validate="many_to_one")
except pd.errors.MergeError as exc:
    print("MergeError:", exc)
```

```
    print("duplicate keys on the right:",
          int(right["product_id"].duplicated().sum()))
```

Expected result: a MergeError naming the violated cardinality, and a count of duplicated keys in the right-hand table.

Verify: this is the error you want. Without validate, the same join would have succeeded and returned a table with more rows than it started with. Fix the right-hand table — usually by drop_duplicates(subset=["[KEY]"]) after establishing which duplicate is correct — rather than removing the validate argument.

Table A.8 sets out the four join types and the circumstances in which each is defensible. The guide's default is the left join, because a fact table enriched with dimension attributes should keep every fact.

Table A.8

Join types and when each is justified

how=	Keeps	Use it when
"left"	Every row of the left table	Enriching a fact table with dimension attributes. The default choice in this guide.
"inner"	Only rows that match on both sides	The analysis is defined only for matched records, and you have counted what the match dropped.
"outer"	Every row of both tables	Reconciling two sources against each other. Rare in the labs; always followed by an unmatched-row report.
"right"	Every row of the right table	Almost never. Swap the tables and use a left join instead, which is easier to read.

A.10 Reshaping Data

Reshaping is another grain question. Before reaching for a function, say what one row of the result should represent. Wide layouts — one row per group, one column per category — read well on a page. Long layouts — one row per observation — are what charts and models expect.

Code A.36. pivot_table: one row per group, one column per category

```
# Input grain: one order line.  Output grain: one month, with one
# column per channel.  Invariant: the grand total is unchanged.
wide = pd.pivot_table(transactions,
                      index="order_month",
                      columns="channel",
                      values="line_revenue",
                      aggfunc="sum",
                      fill_value=0.0,
                      observed=True)

print(wide.round(2))
```

Expected result: a table with one row per month and one column per channel.

Verify: the grand total of the wide table equals the total of the source column. Set fill_value explicitly: an absent combination means no revenue, and leaving it as a missing value will break the arithmetic in the next cell.

Code A.37. crosstab: counts and row shares

```
# lines is the line-grain table with customer attributes joined on,
# built with the validated merge of Code A.33.
lines = transactions.merge(customers[["customer_id", "home_metro"]],
                           on="customer_id", how="left",
```

```
                            validate="many_to_one")

counts = pd.crosstab(lines["home_metro"], lines["channel"])
shares = pd.crosstab(lines["home_metro"], lines["channel"],
                     normalize="index")

print(counts)
print(shares.round(3))
```

Expected result: a table of counts, then the same table as row proportions.

Verify: each row of the normalized table sums to 1. Choose the direction deliberately: normalize="index" answers "of this metro's lines, what share were online?" and normalize="columns" answers "of online lines, what share came from this metro?" They are different questions with different answers.

Code A.38. melt: turn a wide table back into one row per observation

```
long = (wide.reset_index()
        .melt(id_vars="order_month",
              var_name="channel",
              value_name="revenue"))

print(long.head())
```

Expected result: one row per month-and-channel combination, with the value in a single column named revenue.

Verify: the total of the value column equals the grand total of the wide table, and the row count equals rows times columns of the wide table. Charts and models want this shape; a page wants the wide one.

Code A.39. Convert transaction grain to customer grain

```
# Input grain: one order line.  Output grain: one purchasing
# customer, through an order-grain table in between.  Invariants:
# distinct customers, distinct orders, and total revenue.
analysis_date = pd.Timestamp("[ANALYSIS DATE]")

orders = (transactions
          .groupby(["order_id", "customer_id"], as_index=False)
          .agg(order_revenue=("line_revenue", "sum"),
               order_date=("order_date", "min")))

cf = (orders.groupby("customer_id")
      .agg(frequency=("order_id", "nunique"),
           monetary=("order_revenue", "sum"),
           last_order=("order_date", "max"))
      .assign(recency_days=lambda d: (analysis_date
                                      - d["last_order"]).dt.days,
              aov=lambda d: (d["monetary"] / d["frequency"]).round(2))
      .drop(columns="last_order"))

assert len(cf) == transactions["customer_id"].nunique()
assert cf["frequency"].sum() == transactions["order_id"].nunique()
assert np.isclose(cf["monetary"].sum(), transactions["line_revenue"].sum())
assert (cf["recency_days"] >= 0).all()
```

Expected result: one row per customer who purchased, carrying frequency, monetary value, recency in days, and average order value. Four assertions pass silently.

Verify: the four assertions are the point of the cell. Customers equal distinct customers; summed frequency equals distinct orders; summed monetary value equals total revenue; and no recency is negative, which would mean an order dated after the analysis date.

Two steps, not one, and the order matters. Rolling lines straight to customers in a single groupby makes it easy to count order lines where you meant to count orders. Going through an order-grain table first makes each grain change small enough to verify. The customer feature definitions themselves — including the exposure denominator that turns counts into rates — belong to Chapter 6; this pattern is the mechanics only.

A.11 Basic Charts in Python

These are working charts: quick, plain, and made to be looked at by the analyst rather than by an executive. Chapters 12 and 13 govern chart selection, encoding, visual integrity, and communication, and Appendix E is the checklist any chart must pass before it leaves your workspace. Nothing in this section overrides any of that.

Every pattern here uses the object-oriented form — fig, ax = plt.subplots() — rather than the shorter pyplot calls, because the axes object is what you need in order to label anything. And every chart in this guide carries axis labels with units and a title that says what is being shown. An unlabeled chart is not a fast chart; it is an unreadable one.

Every chart in this section is drawn from one frame, order_view: one row per order, carrying order_revenue, units, order_date, and the store_type the order belongs to. Code A.40 builds it, with the validated merge of Code A.33 followed by the grain change of Code A.32; the six entries after it reuse it without rebuilding it. The name is deliberately not orders — Code A.32's orders frame has no store_type, and reusing one name for two different tables is how a notebook starts producing numbers nobody can trace.

> **CONCEPT**
>
> **Aggregate First, Then Plot**
>
> Most misleading charts in student work are misleading before the plotting call. The chart draws exactly what it was handed; the error is in what was handed to it.
>
> So state the grain of the frame you are plotting, aggregate to that grain deliberately, and print the number of rows the chart excludes as missing. A chart that silently drops rows is the visual form of a silent row drop.
>
> Source: Course concept developed for this guide.

That rule applies to every entry below, including the ones whose code does not repeat it. mean, sum, and scatter all skip missing values without saying so, so print the excluded count beside any chart built from a column that can be missing — Code A.41 shows the line, and it is two lines of work in every other entry.

Code A.40. Build the chart frame once, and reconcile it

```
# Every chart below is drawn from order_view. Build it once, from
# the enriched line-grain table of Code A.33: label the digital
# orders rather than leaving them missing, then roll up to orders.
enriched["store_type"] = enriched["store_type"].fillna("Digital")

order_view = (enriched
              .groupby("order_id", as_index=False,
                       dropna=False, observed=True)
              .agg(store_type=("store_type", "first"),
                   order_date=("order_date", "min"),
                   units=("quantity", "sum"),
                   order_revenue=("line_revenue", "sum")))

assert len(order_view) == transactions["order_id"].nunique()
assert np.isclose(order_view["order_revenue"].sum(),
                  transactions["line_revenue"].sum())
```

```
print("orders:", len(order_view), "| revenue:",
      round(order_view["order_revenue"].sum(), 2))
```

Expected result: the order count and the revenue total, with both assertions silent. For the practice file: 12 orders and $1,312.00.

Verify: the grain change preserved the order count and the revenue total. Label the digital orders explicitly instead of leaving store_type missing: a missing group key is dropped by most charting code without comment, and the chart then answers a narrower question than its title claims.

Code A.41. Histogram of one distribution

```
revenue = order_view["order_revenue"].dropna()
print("orders charted:", len(revenue),
      "| orders excluded as missing:", len(order_view) - len(revenue))

fig, ax = plt.subplots(figsize=(7, 4))
ax.hist(revenue, bins=40, edgecolor="white")
ax.set_xlabel("Order revenue ($)")
ax.set_ylabel("Number of orders")
ax.set_title("Distribution of order revenue")
fig.tight_layout()
plt.show()
```

Expected result: a histogram, preceded by a count of what was charted and what was excluded.

Verify: the counts printed above the chart account for every row. Vary the bin count and look again: a shape that appears at 40 bins and disappears at 20 is a property of the bins, not of the data.

Code A.42. Bar chart from a table you aggregated first

```
by_type = (order_view.groupby("store_type")["order_revenue"]
           .mean().sort_values(ascending=False))

fig, ax = plt.subplots(figsize=(7, 4))
ax.bar(by_type.index, by_type.to_numpy())
ax.set_xlabel("Store type")
ax.set_ylabel("Average order value ($)")
ax.set_title("Average order value by store type")
fig.tight_layout()
plt.show()
```

Expected result: one bar per store type, tallest first.

Verify: the bars begin at zero, which matplotlib does by default for bar charts and which Appendix E treats as a stop-ship requirement for ordinary magnitudes. Sorting by the measure is a choice; sorting alphabetically hides the comparison the chart exists to make.

Code A.43. Line chart of an ordered series

```
weekly = (marketing_daily.set_index("date")["revenue"]
          .resample("W").sum())

fig, ax = plt.subplots(figsize=(7, 4))
ax.plot(weekly.index, weekly.to_numpy(), linewidth=0.9)
ax.set_xlabel("Week ending")
ax.set_ylabel("Revenue ($)")
ax.set_title("Weekly revenue")
fig.tight_layout()
plt.show()
```

Expected result: a single line across the whole window.

Verify: the index is sorted and has no gaps: weekly.index.is_monotonic_increasing should be True. A line chart drawn from unsorted dates produces a scribble that looks like volatility.

Code A.44. Scatterplot of two measures

```
fig, ax = plt.subplots(figsize=(7, 4))
ax.scatter(order_view["units"], order_view["order_revenue"], s=8, alpha=0.3)
ax.set_xlabel("Units in the order")
ax.set_ylabel("Order revenue ($)")
ax.set_title("Order revenue against basket size")
fig.tight_layout()
plt.show()
```

Expected result: one point per order.

Verify: one mark is one order, not one line and not one customer. With thousands of points, s and alpha control overplotting; without them a dense cloud hides its own shape.

Code A.45. Box plot by group

```
labels, groups = [], []
for key, part in order_view.groupby("store_type", dropna=False,
                                    observed=True):
    values = part["order_revenue"].dropna()
    print(f"{key}: charting {len(values)}, excluding "
          f"{len(part) - len(values)} as missing")
    labels.append(str(key))
    groups.append(values.to_numpy())

fig, ax = plt.subplots(figsize=(7, 4))
ax.boxplot(groups, whis=1.5)
ax.set_xticks(range(1, len(labels) + 1))
ax.set_xticklabels(labels)
ax.set_ylabel("Order revenue ($)")
ax.set_title("Order revenue by store type")
fig.tight_layout()
plt.show()
```

Expected result: an exclusion count per group, then one box per group in the order the groups appear in labels.

Verify: the tick labels line up with the boxes. Set them explicitly, as here; a box plot whose categories are labeled by position is one reordering away from being wrong.

Code A.46. Add a labeled reference line

```
overall = order_view["order_revenue"].mean()

fig, ax = plt.subplots(figsize=(7, 4))
ax.bar(by_type.index, by_type.to_numpy())
ax.axhline(overall, linestyle="--", linewidth=0.9, color="0.3",
           label=f"All orders: ${overall:,.0f}")
ax.set_xlabel("Store type")
ax.set_ylabel("Average order value ($)")
ax.set_title("Average order value by store type")
ax.legend(fontsize=8)
fig.tight_layout()
plt.show()
```

Expected result: the same bars with a dashed horizontal line and a legend entry naming its value.

Verify: the reference line is labeled with what it represents. An unlabeled line is an unexplained claim, and Appendix E treats it as one. Use axvline for a vertical reference, such as the date a store opened.

Code A.47. Small multiples on shared axes

```
types = sorted(order_view["store_type"].unique())

fig, axes = plt.subplots(1, len(types),
                         figsize=(3.2 * len(types), 3.2),
                         sharex=True, sharey=True)

axes = np.atleast_1d(axes)     # one group still returns a bare Axes

for ax, t in zip(axes, types):
    part = order_view[order_view["store_type"] == t]
    values = part["order_revenue"].dropna()
    print(f"{t}: charting {len(values)}, excluding "
          f"{len(part) - len(values)} as missing")
    ax.hist(values, bins=25, edgecolor="white")
    ax.set_title(f"{t}\nn = {len(values):,}", fontsize=9)
    ax.set_xlabel("Order revenue ($)")

axes[0].set_ylabel("Number of orders")
fig.tight_layout()
plt.show()
```

Expected result: one panel per group, side by side, on identical axes.

Verify: sharex=True and sharey=True are mandatory, not stylistic. Panels on different scales invite exactly the comparison they make invalid. Printing n in each title is what keeps a five-order panel from looking as authoritative as a five-thousand-order one.

Table A.9

Chart command by the question you brought to the data

Question	Command	Entry
How is one measure distributed?	ax.hist, ax.boxplot	A.41, A.45
How do groups compare on one measure?	ax.bar	A.42
How does a measure move over time?	ax.plot	A.43
How do two measures move together?	ax.scatter	A.44
How does this compare with a target or a baseline?	ax.axhline	A.46
Does the pattern hold within every group?	plt.subplots(1, n)	A.46

Table A.9 is a command lookup, and only that. Which form a question actually calls for is not settled here. The reasons these forms work — the perceptual hierarchy that puts position and length above area and angle, the conditions under which a composition chart misleads, the mechanics that make a chart dishonest — are Chapter 12's, and the sign-off any chart must pass is Appendix E's. Consult them before choosing; consult Table A.9 only once the choice is made.

A.12 Common Modeling Workflow Syntax

This section is syntax support and nothing else. It covers the handful of modeling calls that recur across Chapters 7 through 11, so that a student who has already decided what to fit does not have to reopen four chapters to remember the argument order. Every choice these patterns leave open — which features, which model, which threshold, what the coefficients mean — belongs to the chapter named beside it.

CONCEPT

Syntax Here, Interpretation There

A model that runs is not a model that is right. Nothing in this section tells you whether the split respected the unit of prediction, whether a feature leaks the outcome, whether the threshold reflects the real cost of being wrong, or whether the coefficient can bear the verb you want to attach to it.

Chapter 7 owns regression interpretation, Chapter 8 owns the predictive frame and leakage, Chapter 9 owns classification and thresholds, Chapter 10 owns forecasting, and Chapter 11 owns causal claims. Use this section to write the cell; use those chapters to defend it.

Source: Course concept developed for this guide.

Every pattern in this section assumes one frame, model_df: the customer-grain modeling table the chapter builds, indexed by customer_id, carrying the customer_features columns defined in Chapter 6 together with the label the chapter defines — churned for a classification target, spend_next6m for a numeric one. Building that frame is Chapter 8's work, not this appendix's, because the snapshot date, the feature window, the outcome window, and the eligibility rule are analytic decisions rather than syntax. The index matters: every entry below relies on model_df being keyed by customer_id rather than by row position.

Code A.48. Separate features from the target, split once, and freeze it

```
from sklearn.model_selection import train_test_split

assert model_df.index.name == "customer_id"
assert model_df.index.is_unique

FEATURES = ["recency_days", "frequency", "monetary", "aov",
            "tenure_days", "store_share", "app_user", "email_opt_in"]

for forbidden in ["churned", "outcome_orders", "spend_next6m"]:
    assert forbidden not in FEATURES, f"{forbidden} was handed to the model"

train_df, test_df = train_test_split(
    model_df, test_size=0.20, random_state=42,
    stratify=model_df["churned"])       # omit stratify for a numeric target

train_ids, test_ids = train_df.index, test_df.index
assert train_ids.intersection(test_ids).empty
assert len(train_ids) + len(test_ids) == len(model_df)
```

Expected result: two frames whose row counts add to the original, and five assertions that pass silently.

Verify: the two sets share no identifier, and no outcome column appears among the features. Freeze the identities in train_ids and test_ids and reuse them; a second call to train_test_split with the same seed reproduces positions, not customers, so a re-split of a differently ordered frame grades two models on two populations.

The seed is 42 throughout the guide, and the split is 80/20. stratify keeps the outcome's base rate the same in both halves and is used for classification targets; for a numeric target, omit it.

Code A.49. Build a pipeline, fit it, and predict

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, root_mean_squared_error

model = make_pipeline(StandardScaler(), LinearRegression())
model.fit(train_df[FEATURES], train_df["spend_next6m"])
```

```
pred = model.predict(test_df[FEATURES])

print("MAE :", round(mean_absolute_error(test_df["spend_next6m"], pred), 2))
print("RMSE:", round(root_mean_squared_error(
    test_df["spend_next6m"], pred), 2))
```

Expected result: two error figures in the target's own units — dollars, here.

Verify: the prediction array is the same length as the test frame. Any step that learns something from data — a scaler, an imputer, an encoder — belongs inside the pipeline, so that it is refitted within each training fold and never learns a quantity from a customer it is about to be graded on.

Code A.50. Probabilities, and a threshold you can defend

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

# The threshold is an economic decision, derived in Chapter 9 from
# the cost of each kind of error. Never leave it at 0.5 by default.
THRESHOLD = [THRESHOLD FROM CHAPTER 9]

clf = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
clf.fit(train_df[FEATURES], train_df["churned"])

p = clf.predict_proba(test_df[FEATURES])[:, 1]
treat = (p >= THRESHOLD).astype(int)

assert ((p >= 0) & (p <= 1)).all(), "a probability left the unit interval"
print("threshold:", THRESHOLD, "| treated share:", round(treat.mean(), 3))
print("AUC:", round(roc_auc_score(test_df["churned"], p), 3))
```

Expected result: the threshold you supplied, the share of the file it would treat, and the area under the ROC curve.

Verify: predict_proba returns one column per class; [:, 1] takes the probability of the positive class, and taking the wrong column produces a model that appears to be exactly backwards. Never default the threshold to 0.5 without an argument; Chapter 9 derives it from the cost of each kind of error.

Code A.51. Cross-validate every candidate on the same folds

```
from sklearn.model_selection import StratifiedKFold, cross_validate
from sklearn.dummy import DummyClassifier

CV = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

def auc_scorer(estimator, X, y):
    return roc_auc_score(y, estimator.predict_proba(X)[:, 1])

CANDIDATES = {
    "constant floor": (DummyClassifier(strategy="prior"), FEATURES),
    "recency only":   (clf, ["recency_days"]),
    "full logistic":  (clf, FEATURES),
}

rows = []
for name, (est, cols) in CANDIDATES.items():
    s = cross_validate(est, train_df[cols], train_df["churned"],
                       cv=CV, scoring={"auc": auc_scorer})
    rows.append({"candidate": name,
                 "cv_auc": s["test_auc"].mean(),
                 "cv_sd": s["test_auc"].std()})
```

```
board = pd.DataFrame(rows).set_index("candidate").round(3)
print(board)
```

Expected result: a leaderboard, one row per candidate, with a mean score and its spread across folds.

Verify: the constant floor scores about 0.5. If it does not, the scorer or the label is wrong, and every number above it is meaningless. Build the fold generator once and pass the same CV object to every candidate; candidates graded on different folds are not comparable.

Code A.52. An interpretable model with robust standard errors

```
import statsmodels.formula.api as smf

fit = smf.ols(
    "monetary ~ recency_days + frequency + tenure_days"
    " + app_user + email_opt_in + store_share",
    data=train_df).fit(cov_type="HC3")   # see Chapter 7 on robust errors

table = pd.DataFrame({"estimate": fit.params.round(2),
                      "ci_low": fit.conf_int()[0].round(2),
                      "ci_high": fit.conf_int()[1].round(2),
                      "p_value": fit.pvalues.round(4)})

print(table)
print("R-squared:", round(fit.rsquared, 3))
assert (table["ci_low"] <= table["estimate"]).all()
assert (table["estimate"] <= table["ci_high"]).all()
```

Expected result: a coefficient table with an estimate, a confidence interval, and a p-value for each term, plus R-squared.

Verify: every estimate falls inside its own interval — a cheap check that the table was assembled correctly. Use smf.logit in place of smf.ols for a binary target. cov_type="HC3" requests heteroskedasticity-robust standard errors; when that is the right request, and what it changes about the interval, is Chapter 7's.

Code A.53. A results table that joins identifiers, actuals, and predictions

```
scored = (test_df[["churned", "spend_next6m", "recency_days"]]
          .assign(p=p, treated=treat)
          .rename_axis("customer_id")
          .reset_index()
          .sort_values(["p", "customer_id"], ascending=[False, True]))

assert "customer_id" in scored.columns
assert len(scored) == len(test_df)
assert scored["customer_id"].isin(model_df.index).all(), (
    "the index was row positions, not customer identifiers")
print(scored.head().round(3))
```

Expected result: one row per scored customer, highest score first, carrying the identifier, the actual outcome, and the prediction.

Verify: the identifiers are real customers, not row numbers. rename_axis labels whatever index the frame happens to carry, so a frame read without index_col="customer_id" will pass a check for the column name and fail the check above. A ranked list without real identifiers cannot be handed to a campaign, and a score without its actual outcome beside it cannot be evaluated. The second sort key makes the ranking reproducible when scores tie.

Code A.54. A time-aware holdout: never split an ordered series at random

```
series = (marketing_daily.set_index("date").sort_index()
          .asfreq("D")["revenue"])
```

```
# asfreq inserts a row for every absent calendar day, as a missing
# value. Read that gap before deciding what it means.
assert series.notna().all(), (
    f"{int(series.isna().sum())} calendar days have no source row")

HOLDOUT_START = pd.Timestamp("[HOLDOUT START]")
train = series.loc[:HOLDOUT_START - pd.Timedelta(days=1)]
holdout = series.loc[HOLDOUT_START:]

assert train.index.max() < holdout.index.min(), (
    "the training and holdout windows overlap in time")
assert series.index.is_monotonic_increasing
assert series.index.is_unique

seasonal_naive = np.tile(train.iloc[-7:].to_numpy(),
                         int(np.ceil(len(holdout) / 7)))[:len(holdout)]
mae = np.abs(holdout.to_numpy() - seasonal_naive).mean()
print("train days:", len(train), "| holdout days:", len(holdout),
      "| seasonal-naive MAE:", round(float(mae), 2))
```

Expected result: the two window lengths and the baseline's mean absolute error.

Verify: every training date precedes every holdout date, and no calendar day is missing. A random split of a time series lets the model see next week while predicting this one, and the resulting accuracy is fiction. asfreq("D") makes an absent day visible as a gap rather than silently closing it — and a single gap left unhandled turns every error figure computed from the series into nan.

A forecast is graded against a baseline, not against zero. Which baseline, over which horizon, and by what margin a candidate must beat it are Chapter 10's decisions; the tiled repeat above is only the arithmetic that produces one of them.

A.13 Verification Patterns

This is the section to reach for after any transformation, and the one that turns predict-then-verify from a principle into a habit. Each pattern is labeled with the failure it is designed to catch, because a check that catches nothing in particular is decoration.

Two functions do most of the work. assert stops the notebook when a condition you named is false, with a message you wrote; and np.isclose compares floating-point numbers, which are almost never exactly equal even when the arithmetic is right. Use np.isclose for money and np.allclose for arrays of it.

Code A.55. Row counts and key uniqueness

```
rows_before = len(transactions)
working = transactions[transactions["quantity"] > 0].copy()

assert transactions["order_line_id"].is_unique, "order_line_id is not a key"
assert transactions["customer_id"].notna().all(), "a line has no customer"
print("rows:", rows_before, "-> after the filter:", len(working))
```

Expected result: the row count before and after, and silence from both assertions.

Verify: catches a duplicated fact table and orphaned rows. Run the uniqueness check on every fact table before computing anything, and print the count on both sides of every filter.

Code A.56. Totals before and after a transformation

```
total_before = transactions["line_revenue"].sum()
orders = (transactions.groupby("order_id", as_index=False)
          ["line_revenue"].sum())
```

```
total_after = orders["line_revenue"].sum()

assert np.isclose(total_before, total_after), "the roll-up lost revenue"
print(f"{total_before:,.2f} -> {total_after:,.2f}")
```

Expected result: the same figure twice, and silence from the assertion.

Verify: catches revenue lost to a dropped group key or multiplied by a bad join. This is the single most valuable check in the appendix, and it costs two lines.

Code A.57. Allowed categories, date ranges, and bounded shares

```
ALLOWED = {"Online", "App", "Store"}
unexpected = set(transactions["channel"].dropna()) - ALLOWED
assert not unexpected, f"unexpected channel labels: {sorted(unexpected)}"

WINDOW = (pd.Timestamp("[WINDOW START]"),
          pd.Timestamp("[WINDOW END]"))
assert transactions["order_date"].between(*WINDOW).all(), (
    "an order falls outside the declared window")

tol = 1e-9
assert transactions["discount_pct"].between(-tol, 1 + tol).all()
```

Expected result: silence. Any failure names the offending values in its message.

Verify: catches a category variant introduced upstream, a row from outside the window you declared, and a share that is not a share. The tolerance is there because a value summed from rounded components can land a fraction above 1 without being wrong.

Code A.58. Probabilities, split integrity, matrix arithmetic, and shares

```
y = test_df["churned"].to_numpy()      # the actuals, aligned to p

assert ((p >= 0) & (p <= 1)).all(), "a probability left [0, 1]"
assert train_df.index.intersection(test_df.index).empty, (
    "a customer appears in both train and test")

tp = int(((treat == 1) & (y == 1)).sum())
fp = int(((treat == 1) & (y == 0)).sum())
fn = int(((treat == 0) & (y == 1)).sum())
tn = int(((treat == 0) & (y == 0)).sum())
assert tp + fp + fn + tn == len(y), "the confusion matrix must sum to n"

shares = scored.groupby("treated").size() / len(scored)
assert np.isclose(shares.sum(), 1.0)
print(f"TP {tp}  FP {fp}  FN {fn}  TN {tn}  n {len(y)}")
```

Expected result: the four cell counts and the total, with every assertion silent.

Verify: catches a score column that is not a probability, a leaked test customer, a confusion matrix built from mismatched arrays, and a set of shares that do not partition the file.

Code A.59. Compare the code against a hand calculation

```
# By hand from the practice file, order O10003:
#   88.00 + 84.00 + 76.00 = 248.00
BY_HAND = 248.00

computed = transactions.loc[transactions["order_id"] == "O10003",
                            "line_revenue"].sum()
```

```
assert np.isclose(computed, BY_HAND), f"{computed} != {BY_HAND}"
print(f"computed {computed:,.2f}   hand {BY_HAND:,.2f}   agree")
```

Expected result: the two figures side by side, agreeing.

Verify: catches an error in the logic itself, which no internal consistency check can find. Do this once on a miniature you can compute with a calculator before running the same code on eighty thousand rows.

Table A.10 collects every check in this section and names the failure each is designed to catch, so that a check can be chosen by the risk you are worried about rather than by the command it uses.

Table A.10

Verification patterns and the failure each is designed to catch

Check	Failure it catches	Entry
Row count before and after	A filter or join that silently changed the population	A.55
Key uniqueness	A duplicated fact table inflating every total	A.55
Total before and after	Revenue lost in a roll-up or multiplied by a join	A.56
Row count unchanged by a merge	A many-to-many join expanding the table	A.33
Unmatched-key report	Dimension rows that do not exist, becoming missing values	A.34
Allowed-category set	A label variant introduced upstream	A.57
Declared date window	Rows from outside the period the claim covers	A.57
Bounded shares	A ratio computed on the wrong denominator	A.57
Probabilities within [0, 1]	The wrong predict_proba column, or a score mistaken for a probability	A.58
Train and test identifiers disjoint	A leaked customer flattering the test score	A.58
Confusion matrix sums to n	Actuals and predictions misaligned	A.58
Shares sum to 1	A partition that does not partition	A.58
Agreement with a hand calculation	An error in the logic that every internal check would pass	A.58

The evidence these checks produce is not only for you. Appendix D's verification table asks for the claim checked, the prediction made before running, the method, the expected value, the actual result, and the action taken — which is exactly what a printed row count, a reconciled total, or a passing assertion supplies. Predict the number first, then run the check; a verification recorded after the fact verifies nothing.

A.14 Common Errors and Troubleshooting

Organized by symptom, because that is what you have when something goes wrong. Read the last line of the traceback first, find the symptom below, and work through the four steps: what it usually means, what to inspect first, a safe next step, and what not to do.

Code A.60. Read the last line first

```
try:
    result = transactions["quantity"] + transactions["channel"]
except TypeError as exc:
    print("last line of the traceback:")
    print(f"  TypeError: {exc}")
    print("types involved:",
          transactions["quantity"].dtype, transactions["channel"].dtype)
```

Expected result: the error type, its specific complaint, and the two column types responsible.

Verify: the error type names the family of the problem and the message names the instance. Printing the dtypes of the columns involved resolves most TypeError cases in one line.

A.14.1 FileNotFoundError, or No such file or directory

Usually means the file is not where the notebook is looking, or the name is spelled differently — file names are case-sensitive, and a trailing space in a name is invisible. Inspect first with Code A.5, which lists what the runtime can actually see. Safe next step: reload the file from the companion location using Code A.2, or re-upload it with Code A.3 if the runtime restarted. Do not retype the path from memory into six cells; set DATA once and build every path from it.

A.14.2 KeyError, or a column that is not found

Usually means a spelling difference, a case difference, or a column that an earlier cell renamed or dropped. Inspect first with list(df.columns). Safe next step: correct the name, or rerun the cell that creates the column, then re-run from the top so the notebook state matches the notebook text. Do not create a new column with the name you expected in order to make the error disappear; that produces a table with two versions of the same field.

A.14.3 TypeError or ValueError during arithmetic

Usually means a column that should be numeric is stored as text, typically because it arrived with a currency symbol or a thousands separator. Inspect first with df.dtypes and df["[COLUMN]"].head(). Safe next step: coerce with Code A.26 and compare the missing-value count before and after. Do not wrap the operation in a conversion inside every cell; fix the column once, near the top.

A.14.4 ValueError: Length of values does not match length of index

Usually means you are assigning an array built from a filtered or reordered frame back onto the full one. Inspect first by printing the length of the frame and of the value you are assigning. Safe next step: assign onto the same frame the values were computed from, or align on the index with .reindex before assigning. Do not use .to_numpy() or .values to silence the complaint — that discards the index alignment that was protecting you.

A.14.5 ModuleNotFoundError

Usually means the library is not present in the runtime, which in Colab is unusual for pandas, NumPy, Matplotlib, scikit-learn, and statsmodels and common for anything else. Inspect first by re-running the import cell alone. Safe next step: install it in a cell with !pip install [PACKAGE] and re-run the imports; record the install in the notebook so the next reader sees it. Do not install a package the assignment did not ask for in order to avoid writing a few lines of pandas.

A.14.6 NameError, after a runtime restart

Usually means the variable was defined in a cell that has not been run in this session. The notebook text and the runtime memory have drifted apart. Inspect first by checking whether the Connect indicator shows a fresh runtime. Safe next step: Runtime → Restart and run all, which is the only state you should ever submit. Do not hunt for the one cell to re-run; that leaves the drift in place for the next reader to discover.

A.14.7 A merge that produced more rows than it started with

Usually means duplicated keys in the right-hand table, so each left row matched several right rows. Inspect first with right["[KEY]"].duplicated().sum() and with the row count on both sides. Safe next step: resolve the duplicates in the right-hand table, then re-run the merge with validate="many_to_one" as in Code A.33. Do not drop the extra rows afterward with drop_duplicates; that keeps an arbitrary one of several matches and hides the real problem.

A.14.8 A blank or nearly empty chart

Usually means the frame handed to the plotting call is empty, or the column being plotted is entirely missing values, or a filter earlier in the cell removed everything. Inspect first by printing len(frame) and

frame["[COLUMN]"].notna().sum() immediately before the plot. Safe next step: fix the upstream filter and print the excluded count, as Code A.41 does. Do not adjust the axis limits to make something appear.

A.14.9 Dates that sort or group strangely

Usually means the column is still text. ISO-formatted text such as 2025-12-31 may appear to sort chronologically, but that is a property of the format — largest unit first — rather than evidence that the column is a date; 12/31/2025 and 31-Dec-2025 sort wrongly outright, and no text column supports date arithmetic, resampling, or the .dt accessor. Inspect first with df.dtypes: a date column reports a datetime type, not a text one. Safe next step: parse at read time with parse_dates, or convert with Code A.27 and confirm that no value failed to parse. Do not infer from one well-behaved sort that the column is a date.

A.14.10 A model that refuses text or missing values

Usually means a categorical column reached the estimator without being encoded, or a feature contains missing values that scikit-learn will not silently fill. Inspect first with train_df[FEATURES].dtypes and train_df[FEATURES].isna().sum(). Safe next step: encode the categorical explicitly as indicator columns and handle the missing values as a documented decision, inside the pipeline. Do not call .dropna() on the feature frame without counting and reporting what it removed; a model fitted on the rows that happened to be complete is a model fitted on a different population.

A.14.11 Model performance that looks too good

Usually means leakage: a feature that encodes the outcome, a transformation fitted before the split, or a customer appearing on both sides of it. This is the most consequential entry in this section, because nothing about it looks like an error. Inspect first by listing every feature and asking, for each, whether its value could have been known at the moment of prediction; then re-run the checks in Code A.48 and Code A.58. Safe next step: remove the suspect feature and refit — a real signal survives, a leak collapses. Do not report the number. Chapter 8 treats an unexpectedly excellent result as a red flag rather than an achievement, and so should you.

A.15 Saving, Exporting, and Submitting Work

The last thing that goes wrong in an assignment is usually not the analysis. It is a notebook that will not run for anyone else, a figure exported at a size no one can read, or a file that never made it off the temporary machine.

Code A.61. Export a table and a figure

```
summary = (transactions.groupby("channel", as_index=False)
           .agg(orders=("order_id", "nunique"),
                revenue=("line_revenue", "sum")))
summary.to_csv("channel_summary.csv", index=False)

fig, ax = plt.subplots(figsize=(7, 4))
ax.bar(summary["channel"], summary["revenue"])
ax.set_xlabel("Channel")
ax.set_ylabel("Revenue ($)")
ax.set_title("Revenue by channel")
fig.tight_layout()
fig.savefig("revenue_by_channel.png", dpi=200, bbox_inches="tight")
```

Expected result: two files in the working directory, and no output in the notebook.

Verify: read the CSV back and confirm its row count and total; open the PNG and confirm the axis labels are legible at the size it will be used. dpi=200 is the guide's minimum for a figure that will be placed in a document, and bbox_inches="tight" keeps the labels from being cut off.

Code A.62. Confirm the notebook reproduces from a clean runtime

```
EXPECTED_TOTAL = 1312.00      # replace with your own certified total

required = ["order_id", "customer_id", "order_date", "line_revenue"]
missing = [c for c in required if c not in transactions.columns]
assert not missing, f"columns absent from the file: {missing}"

assert np.isclose(transactions["line_revenue"].sum(), EXPECTED_TOTAL), \
    "the certified total no longer reproduces"

print("STATUS: reproduces from a clean runtime.")
```

Expected result: one status line, after Runtime → Restart and run all completes with no error.

Verify: the check has to be run after a restart, not after an ordinary execution. A notebook that only works in the order you happened to run the cells is not a notebook anyone else can use, including you next week.

A.15.1 The Submission Checklist

35. Runtime → Restart and run all completes with no error, top to bottom.
36. The environment record from Code A.12 is near the top of the notebook.
37. Every transformation cell prints or asserts its verification, and the verification log is complete.
38. Every AI exchange is documented per Appendix D, including at least one output you corrected and the evidence you used.
39. Outputs are retained or cleared according to the assignment instructions; when in doubt, retain them.
40. No credentials, API keys, personal identifiers, or confidential data appear anywhere in the notebook, including in a commented-out cell.
41. Figures and tables the assignment asks for are exported as files, at a readable size.
42. The file is named so a reader knows whose it is and which assignment it answers.
43. You have downloaded the notebook and any exported files off the runtime, and submitted the files themselves rather than a screenshot of them.

Table A.11 sets out which export format is right for which purpose. The notebook itself is the submission; the other formats are supporting material.

Table A.11

Export formats and when each is the right one

Format	Use it for	Note
.ipynb	The submitted notebook	Preserves code, output, and text together. The default.
.py	A script the course explicitly asks for	Loses all output and all text cells.
PDF	A read-only record when the assignment requires one	A picture of the work, never a substitute for it.
.csv	A table another tool will read	Always with index=False.
.png	A figure for a document or a slide	dpi=200 or higher, bbox_inches="tight".

A.16 Command Index by Task

This index is arranged by what you are trying to do rather than by what the command is called, so that it is usable before you know the function's name. The entry column points to the code entry where the pattern appears in context, with its expected result and its verification.

Table A.12 is that index. Where a task has more than one entry, both are listed, and the earlier one is usually the simpler.

Table A.12

Command index, by task

To do this	Use	Entry
Check where the runtime is and what it can see	`os.getcwd(), Path().glob()`	A.1, A.5
Load a course CSV with dates parsed	`pd.read_csv(..., parse_dates=[...])`	A.2
Upload a file from your computer	`files.upload()`	A.3
Use a file in Google Drive	`drive.mount()`	A.4
Write a table out	`.to_csv(..., index=False)`	A.6, A.61
See how many rows and columns arrived	`.shape`	A.13
See the first rows	`.head(), .tail()`	A.13
Check that each column is the right type	`.dtypes, .info()`	A.14
Summarize the numeric columns	`.describe()`	A.14
Count missing values	`.isna().sum()`	A.15, A.23
List the labels in a category	`.value_counts(dropna=False)`	A.15
Confirm a column is a key	`.is_unique, .nunique()`	A.16
Keep some columns	`df[["a", "b"]]`	A.17
Keep some rows	`df[condition], .isin()`	A.18
Sort reproducibly	`.sort_values([measure, id])`	A.19
Build a derived column	`df["new"] = ...`	A.20
Rename a column	`.rename(columns={...})`	A.21
Recode categories	`.map(dictionary)`	A.21
Work on a subset safely	`.copy()`	A.22
Fill or drop missing values	`.fillna(), .dropna(subset=[...])`	A.24
Find duplicates	`.duplicated(subset=[...])`	A.25
Turn currency text into numbers	`pd.to_numeric(..., errors="coerce")`	A.26
Turn text into dates	`pd.to_datetime(..., errors="coerce")`	A.27
Pull year, month, or weekday out of a date	`.dt.year, .dt.to_period("M"), .dt.day_name()`	A.27
Summarize by group	`.groupby(...).agg(name=(col, func))`	A.29
Count distinct orders or customers	`("order_id", "nunique")`	A.29, A.31
Compute shares that sum to 1	`col / col.sum()`	A.31
Roll lines up to orders	`.groupby("order_id").agg(...)`	A.32
Join a dimension onto a fact table	`.merge(..., validate=, indicator=)`	A.33
Find keys that did not match	`set(left) - set(right)`	A.34
Go wide	`pd.pivot_table()`	A.36
Cross-tabulate two categories	`pd.crosstab(..., normalize=)`	A.37
Go long	`.melt()`	A.38

To do this	Use	Entry
Build a customer-grain table	`two-step groupby`	A.39
Draw a distribution	`ax.hist(), ax.boxplot()`	A.41, A.45
Compare groups	`ax.bar()`	A.42
Show a trend	`ax.plot()`	A.43
Show a relationship	`ax.scatter()`	A.44
Add a baseline to a chart	`ax.axhline(), ax.axvline()`	A.46
Repeat a chart per group	`plt.subplots(1, n, sharey=True)`	A.47
Split into train and test	`train_test_split(..., stratify=)`	A.48
Scale and fit in one object	`make_pipeline()`	A.49
Predict a number	`.predict()`	A.49
Predict a probability	`.predict_proba()[:, 1]`	A.50
Compare candidates on identical folds	`cross_validate(..., cv=CV)`	A.51
Fit a readable regression	`smf.ols(...).fit(cov_type="HC3")`	A.52
Build a scored list for a campaign	`.assign(p=...).reset_index()`	A.53
Hold out the end of a time series	`.loc[:cut], .loc[cut:]`	A.54
Stop the notebook when a check fails	`assert condition, "message"`	A.55
Compare two money totals	`np.isclose(), np.allclose()`	A.56
Save a figure	`fig.savefig(..., dpi=200)`	A.60

References

The library documentation is authoritative for every argument named in this appendix, and the works below are the citable references for the packages introduced in Table A.4 and for the Colab environment described in Section A.1. This appendix records the recurring subset the guide uses; it does not replace the reference manuals.

Google. (n.d.). *Google Colaboratory: Frequently asked questions*. Retrieved August 2, 2026, from Google Colab frequently asked questions

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 92–96).

Appendix B

Common Marketing Metrics

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.1 · August 2026
Except where otherwise noted, this appendix is licensed under CC BY 4.0.

Appendix Information

PURPOSE

This appendix is the guide's metric dictionary. It collects, in one place and in one format, every quantity the chapters and the two projects compute, so that a metric is fully defined once rather than defined differently in several places. Each entry states the business question the metric answers, its formula, its numerator and denominator, its window, its filters, its grain, its unit, what a higher or lower value means, the mistake most likely to be made with it, and a check it must survive.

USE THIS APPENDIX WHEN

Consult this appendix when you are about to compute a named metric, when two figures bearing the same name disagree, when you need to state a definition in a notebook, a dashboard footer, or a project methods section, or when you are checking whether a number someone has handed you means what its label says.

THIS APPENDIX DOES NOT

This appendix is a definition catalog. It does not teach the methods that produce the measures, does not attempt to catalog every metric used in marketing practice, and does not treat every metric as a key performance indicator. It catalogs only what the guide, the labs, the exercises, and the two projects actually use; Section B.14 records what was considered and left out, and why. Where a metric raises a question of method rather than definition, the entry names the chapter that owns the decision and stops there.

VERSION AND DATE

Version 1.1 · August 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Common marketing metrics. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Appendix B, Version 1.1) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Except where otherwise noted, this appendix is licensed under a Creative Commons Attribution 4.0 International License.
Copyright © 2026 by Jose Mendoza.

Google Colab is a product of Google LLC. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas is a sponsored project of NumFOCUS, a 501(c)(3) nonprofit charity in the United States. Tableau is a trademark of Salesforce, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this appendix, including outlining, preliminary drafts, revision of prose, support for code and analytical examples, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the appendix's scope, boundaries, and content, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and every formula and worked example was independently recomputed using either the appendix practice file or the published figures of the chapter that owns the metric, with each stated result confirmed. Responsibility for the accuracy, originality, and final form of this appendix rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

COMPANION FILES

A machine-readable metric dictionary carrying every field of every entry below, a printable metric-definition form, tested Python functions for the most repeated metrics, a changelog, and the practice file these worked examples run on are in the Applied Business Analytics companion repository.

How to Read an Entry

Entries in this appendix are organized by metric family, not alphabetically, because the mistake this appendix exists to prevent is confusing two metrics that sit near each other rather than failing to find one. Section B.14 supplies the alphabetical index and names the metrics that were considered and deliberately excluded.

Every entry uses the same fields, in the same order. Table B.1 lists them. The first six are the elements that decide what number comes out; the last five are what a reader needs in order to use the number without misreading it. A field that a source genuinely does not specify is not left blank and is not guessed at; it is the analyst's to decide and to record.

Table B.1

The fields every metric entry carries, and what each one settles

Field	What it settles
Business question	The decision or comparison the metric supports. A metric with no decision behind it cannot be judged fit for anything.
Formula	Plain language first, then the code or symbolic form where precision requires it.
Numerator	The exact event, amount, or count included.
Denominator	The exact eligible base. For a total rather than a rate the field reads "none", which is itself information.
Window	The period and the boundary rule. A trailing window needs a start and an as-of date.
Filters	Channel, customer, campaign, product, and eligibility rules, applied identically to numerator and denominator.
Grain	What one computed observation represents. This is the fifth element the four-element discipline of Section 3.5.1 leaves implicit, and the one Appendix A promises is stated here.
Unit	Dollars, count, proportion, percentage, percentage points, days, or ratio. Percentage and percentage points are different units.
Required fields	The minimum data elements the computation needs.
Interpretation	What a higher or lower value means, and what the metric does not establish.
Common failure	The definitional or interpretive mistake most likely to be made.
Verification	A hand check, a reconciliation, or a bound the metric must satisfy.
Common variants	Present only where practice legitimately differs, so that a defensible alternative is named rather than silently overruled.
Owned by	The chapter that introduces and teaches the measure.

CONCEPT

One Definition, In One Place

A metric exists only as its definition. Two analysts who apply different definitions to the same data are not making an arithmetic error; they are computing different quantities, and no amount of recomputation will reconcile them. This appendix is the guide's single copy of each definition, and code, dashboards, and prose implement the entry rather than inventing a second version of it.

That rule has a consequence worth stating plainly. When a chapter, a lab, or a project needs a metric this appendix does not carry, the answer is not to invent one locally. It is to write the entry, using the form in Section B.1, and record it where the people who will disagree can see it.

> Source: Course concept developed for this guide, informed by Farris et al. (2010) and Kimball and Ross (2013).

Worked examples in this appendix run on the same practice file that ships with Appendix A: twenty order lines, twelve orders, eight customers, and $1,312.00 of revenue, shaped exactly like StyleCraft's transactions table and small enough to check with a calculator. Where a family has no natural miniature in that file — media, funnel, classification, forecasting, and experiments — the example reuses the figures the owning chapter already publishes, so that a number quoted here can be traced back to the page it came from. No number in this appendix answers a graded exercise.

B.1 How to Use This Metric Dictionary

Chapter 3 established that a number is a claim: every reported figure is shorthand for a longer sentence naming a definition, a unit of analysis, a window, and a set of filters. This appendix writes that longer sentence down, once, for every metric the guide uses. This section explains how to read the catalog, how to cite an entry, and what to do when a project needs a metric the catalog does not carry.

B.1.1 A metric is not a KPI

A metric is a quantity computed from data by an explicit rule, at a stated grain, for description or comparison. A key performance indicator is a metric an organization has elevated: attached to a target, assigned an owner, reviewed on a cadence, and connected to a decision (Parmenter, 2015). The distinction is not pedantry. An organization can compute hundreds of metrics and attend to only a handful, and calling every metric a KPI inflates a dashboard until nothing on it is key. Elevation also carries a cost this appendix cannot remove: once a measure becomes a target, behavior reorganizes around the measure and the measure degrades as evidence (Campbell, 1979). And because data quality is fitness for a use rather than a property of a file, a metric with no decision behind it cannot be judged fit for anything (Wang & Strong, 1996).

Every entry in Sections B.2 through B.12 is a metric. Section B.13 is different in kind: it catalogs the measures that exist only because a decision elevated them — a coverage threshold, a fixed-cost line, a forecast interval expressed in store-equivalents — and it explains the apparatus by which any entry in this appendix becomes a dashboard KPI. It creates no new formulas.

B.1.2 Five elements, not four

Section 3.5.1 requires that a ratio metric name its numerator, its denominator, its time window, and its filters. This appendix adds a fifth element to every entry, ratio or not: the grain. The addition is not a refinement of Chapter 3's rule but a consequence of it, and Appendix A commits to it by name — its entry for average order value points here precisely because the grain is stated here.

The reason is that a denominator often conceals a grain rather than revealing it. Average order value divides revenue by orders, and the metric's grain is the order; return on ad spend divides revenue by dollars of spend, and its grain is the campaign, not the dollar. Where the two come apart, an analyst who has checked only the four elements can still compute a defensible-looking number at the wrong grain. Chapter 12 reports the consequence in its most expensive form: an average line value of $50.41 and a true average order value of $108.00, on the same file, with no error message and no visual difference between the correct chart and the incorrect one.

B.1.3 Why two correct formulas disagree

The commonest failure this appendix is designed to prevent is not miscalculation. It is two people computing different quantities under one name and then arguing about arithmetic. Chapter 3 stages the case: three StyleCraft teams report repeat purchase rates of 60, 43, and 20 percent, and none of them has made a mistake. Table B.2 reproduces the three definitions from the eighteen-order preview the chapter computes them on, and shows where they part company.

Table B.2

Three correct repeat purchase rates, and the elements that separate them

Element	Lifetime (Retail)	Trailing 12 months (CRM)	Store only (Finance)
Numerator	Customers with 2+ orders ever	Customers with 2+ orders in the window	Customers with 2+ store orders in the window
Denominator	All customers with 1+ completed order in the available history	Customers with 1+ order in the window	Customers with 1+ store order in the window
Window	Entire history to the analysis date	Trailing 12 months to the analysis date	Trailing 12 months to the analysis date
Filters	All channels	All channels	Store channel only
Grain	Customer	Customer	Customer
Numerator count	6	3	1
Denominator count	10	7	5
Value	0.60	0.43	0.20

Read across the value row and the dispute looks like a data-quality problem. Read down the denominator row and it stops being mysterious: the three rates are computed over ten, seven, and five customers respectively, because each definition admits a different population. The organization does not need to decide which number is true. It needs to decide which definition serves the decision, record it, and report the others beside it under their own names. Section B.4's entry for repeat purchase rate carries the outcome of that decision, with the two alternatives named as variants.

B.1.4 Formula and notation standards

Every entry follows the same conventions, and a metric defined outside this appendix should follow them too.

Plain language leads. Each formula is stated in words first and in symbols or code only where precision requires it. A formula a stakeholder cannot read aloud is a formula nobody will audit.

The unit is named, and percentage points are not percentages. A rate moving from 20 to 24 percent rose by four percentage points and by twenty percent. Both are true and they are not the same claim; Chapter 13 names the practice of choosing between them for effect as framing.

Rates are recomputed from pooled totals, never averaged. The mean of group means is not the overall mean unless the groups are the same size. Where the unweighted average is genuinely wanted, it is labeled as one and its difference from the pooled figure is reported.

A rate over an empty base is undefined, not zero and not infinite. A campaign with no clicks has no cost per click. A table of infinities is a denominator problem wearing a printing error.

Rounding is a display decision and never a computational one. Formulas carry no implied rounding; figures are rounded when they are shown, and the rounding is not carried back into a subsequent calculation.

A classification metric travels with its base rate; a growth metric travels with its comparison period; an attributed metric travels with its attribution window. Each of these is the element without which the number cannot be read at all.

Causal verbs are earned, not chosen. Entries state what a measure does not establish. An attributed return is not an incremental one, a predicted probability is not a treatment effect, and an adjusted difference is not a causal effect.

B.1.5 Citing a definition, and adding one

A metric quoted outside this appendix carries a pointer to it: in a notebook, a comment naming the entry; in a dashboard, a footer naming the source, window, filters, and grain; in a project methods section, the entry key and the version of this appendix. The companion dictionary ships the same fields in machine-readable form, which is what an AI assistant should be given as context rather than being asked to supply a definition of its own.

The discipline is not peculiar to this guide. A public company presenting a metric is expected to define it, to say why it is useful and how management uses it, and to disclose the effects of any change in how it is calculated (U.S. Securities and Exchange Commission, 2020). A project deliverable owes its reader the same three things.

When a project needs a metric this appendix does not carry, do not modify an entry. Write a new one on the form in Table B.3, record it in the project's own dictionary, and note which existing entry it resembles and how it differs. A project-specific metric that quietly redefines a cataloged one is the Chapter 3 dispute reproduced at project scale.

Table B.3

The metric definition form, for a metric this appendix does not carry

Field	Your entry
Metric name	
Business question	
Formula	
Numerator	
Denominator	
Window	
Filters	
Grain	
Unit	
Required fields	
Interpretation	
Common failure	
Verification	
Owner and decision use	
Nearest cataloged entry, and how this differs	

B.1.6 The practice file the examples run on

Worked examples in Sections B.2 through B.4 and B.8 run on the practice file described above, rolled up from twenty order lines to the twelve orders in Table B.4. Every figure in those examples can be recovered from this table with a calculator, which is the point of keeping it small. The digital orders carry the reserved store identifier ONLINE and therefore have no store type, which is why they appear as a third group rather than being dropped.

Table B.4

The practice file at order grain, with its certified totals

Order	Customer	Store type	Revenue	Cost	Units	Lines
O10001	C001	Urban	96.00	48.00	2	2
O10002	C003	Digital	117.00	63.00	3	2
O10003	C002	Suburban	248.00	124.00	3	3
O10004	C004	Digital	84.00	42.00	2	1
O10005	C001	Urban	78.00	39.00	2	1
O10006	C005	Digital	76.00	44.50	2	2
O10007	C002	Suburban	192.00	96.00	3	3
O10008	C006	Digital	68.00	34.00	2	1
O10009	C001	Urban	60.00	30.00	2	1
O10010	C003	Digital	112.00	64.00	2	2
O10011	C007	Digital	88.00	44.00	2	1
O10012	C008	Digital	93.00	46.50	3	1
Total	**8 customers**		**1,312.00**	**675.00**	**28**	**20**

B.2 Revenue, Margin, and Volume Metrics

Every metric in this family is an amount or a count rather than a rate, which makes them the easiest to compute and among the easiest to misreport. Their failure mode is not a wrong denominator but an unstated one: a revenue total whose window, filters, and returns treatment nobody wrote down cannot be reconciled against a second total that used different ones.

B.2.1 Line revenue

Question. What did one order line contribute to net sales? **Formula.** quantity × unit_price × (1 − discount_pct); stored as line_revenue and recomputed rather than trusted.

Numerator. The extended, discounted value of a single order line. **Denominator.** None — it is an amount, not a rate. **Window.** The order's order_date; no window of its own. **Filters.** Certified lines only; a line whose quantity, unit_price, or discount_pct is unresolved has no line revenue and must stay missing rather than becoming zero. **Grain.** One order line. **Unit.** Dollars. **Required fields.** quantity, unit_price, discount_pct.

Interpretation. The building block every revenue metric in this appendix aggregates. It is net of line-level discount and gross of returns, credits, and refunds. **Common failure.** Summing a column in which unresolved lines were filled with 0.00. pandas sum() skips missing values by default, so a total computed over eleven lines of which one is unresolved is the revenue of ten lines wearing the label of eleven. **Verification.** Recompute from quantity, unit_price, and discount_pct and compare with the stored column; then confirm the count of unresolved lines is zero before quoting any total.

Common variants. Where returns are netted at line level, the same name denotes a different quantity. State whether returns and adjustments are inside the number.

Owned by. Chapter 4, Section 4.8.

B.2.2 Gross sales

Question. What would the period have billed at catalog prices, before discount? **Formula.** Sum of quantity × unit_price across the included lines.

Numerator. Extended catalog value of every included line. **Denominator.** None — it is a total. **Window.** Stated period, both ends inclusive. **Filters.** Stated channel, store, and product filters. **Grain.** Whatever population is stated; commonly chain, store, or campaign. **Unit.** Dollars. **Required fields.** quantity, unit_price.

Interpretation. Read only beside net revenue. Alone it overstates what the business earned by exactly the discount given. **Common failure.** Reporting gross sales as 'revenue'. The gap between the two names is the whole promotional story, and collapsing it hides the discount dependence Chapter 6 measures. **Verification.** Gross sales less discount dollars must equal net revenue exactly.

Owned by. Appendix B; the book uses net revenue throughout.

B.2.3 Discount dollars

Question. How much price was given away to produce this revenue? **Formula.** Gross sales − net revenue; equivalently sum of quantity × unit_price × discount_pct.

Numerator. Catalog value forgone on the included lines. **Denominator.** None — it is a total. **Window.** Stated period. **Filters.** Stated filters; lines with discount_pct of 0 contribute nothing. **Grain.** Stated population. **Unit.** Dollars. **Required fields.** quantity, unit_price, discount_pct.

Interpretation. Rising discount dollars against flat units is a margin story, not a demand story. **Common failure.** Confusing discount dollars with markdown at retail, which is an inventory-accounting quantity computed against a different base. **Verification.** Must be non-negative, and must equal gross sales less net revenue.

Owned by. Appendix B.

B.2.4 Net revenue

Question. What did the business actually earn from these orders? **Formula.** Sum of line_revenue across the included lines.

Numerator. Sum of line_revenue. **Denominator.** None — it is a total. **Window.** Stated period, both ends inclusive; a trailing window needs a start and an as-of date. **Filters.** Stated channel, store, customer, and campaign filters. **Grain.** Stated population — chain, store, metro, customer, campaign, or day. **Unit.** Dollars. **Required fields.** line_revenue, order_date, plus whatever the filters name.

Interpretation. The book's default meaning of the bare word revenue. It is net of line discount and does not net returns. **Common failure.** Quoting a revenue total whose window, filters, and returns treatment are unstated, so two correct totals disagree and neither can be reconciled. **Verification.** Reconcile to the certified source total to the cent. An aggregation redistributes rows and must never create or destroy dollars.

Common variants. Recognized revenue as finance defines it may net returns, cancellations, and gift-card breakage. Where the dashboard sits beside a finance report, say which is being shown.

Owned by. Chapter 4, Section 4.8.

B.2.5 Line cost

Question. What did the goods on one order line cost to sell? **Formula.** quantity × unit_cost.

Numerator. Extended variable cost of a single order line. **Denominator.** None — it is an amount. **Window.** The order's order_date. **Filters.** Certified lines only. **Grain.** One order line. **Unit.** Dollars. **Required fields.** quantity, unit_cost.

Interpretation. What line_cost carries decides whether the difference below it is contribution or gross profit. In the certified extract it carries merchandise cost, inbound freight, payment processing, and pick-and-ship. **Common failure.** Assuming the cost column is fully loaded when it carries merchandise cost alone. If it does, every quantity below is gross profit and must be relabeled. **Verification.** Confirm with finance what line_cost contains before naming anything computed from it.

Owned by. Chapter 13, Section 13.12.1.

B.2.6 Contribution

Question. What is left after the variable cost of selling, before the fixed footprint? **Formula.** Sum of (line_revenue − line_cost).

Numerator. Net revenue less variable cost on the included lines. **Denominator.** None — it is a total. **Window.** Stated period. **Filters.** Stated population; the go/no-go dashboard computes it for suburban and resort stores only. **Grain.** Stated population; the dashboard uses one store-week. **Unit.** Dollars. **Required fields.** line_revenue, line_cost.

Interpretation. Revenue after variable cost and before the fixed footprint. It is the quantity a store must generate enough of to cover its own rent. **Common failure.** Calling it contribution when line_cost carries merchandise cost alone. That quantity is gross profit, and every sentence using the word contribution is then wrong. **Verification.** Must be less than net revenue on the same population. Recompute on one store-week by hand from the underlying lines.

Common variants. Contribution after occupancy, and contribution after allocated overhead, are different and larger deductions. Name the deduction, not just the word.

Owned by. Chapter 13, Section 13.12.1.

B.2.7 Contribution margin

Question. What share of each revenue dollar survives variable cost? **Formula.** Sum of (line_revenue − line_cost) / Sum of line_revenue, on the same cost definition.

Numerator. Contribution on the stated population. **Denominator.** Net revenue on the same population. **Window.** Stated period. **Filters.** Same population for numerator and denominator, without exception. **Grain.** One population, not one line — it is a pooled ratio. **Unit.** Percentage. **Required fields.** line_revenue, line_cost.

Interpretation. A within-population conversion factor. Chapter 13 computes 46 percent for the suburban and resort stores, which is not the chain's margin and may not be used as one. **Common failure.** Applying a margin computed on one population to a quantity measured on another. The chain's contribution and the suburban stores'

margin do not belong in the same fraction. **Verification.** Bounded above by 1. Recompute from the two totals rather than averaging line-level margins, which weights a $24 line like a $248 one.

Common variants. Gross margin uses merchandise cost alone in the numerator and is systematically larger. The two are routinely swapped in decks.

Owned by. Chapter 13, Section 13.12.1.

B.2.8 Net contribution after advertising spend

Question. Did this campaign's revenue survive both product cost and the advertising that produced it? **Formula.** revenue × margin_rate − ad_spend.

Numerator. Gross margin dollars on campaign revenue, less advertising spend. **Denominator.** None — it is an amount. **Window.** The campaign window. **Filters.** One campaign; advertising spend only, excluding creative production and agency fees. **Grain.** One marketing campaign. **Unit.** Dollars. **Required fields.** revenue, margin_rate, ad_spend.

Interpretation. The correction to return on ad spend that margin supplies. A campaign can lead on return on ad spend and still destroy contribution. **Common failure.** Calling it contribution margin. It is not: it excludes every marketing cost other than advertising, and the name is chosen so the two cannot be confused. **Verification.** Recompute one campaign by hand. Confirm the sign: a negative value is a finding, not an error.

Owned by. Chapter 1, Lab 1.2.

B.2.9 Units sold

Question. How much merchandise moved, independent of price? **Formula.** Sum of quantity across the included lines.

Numerator. Sum of quantity. **Denominator.** None — it is a count. **Window.** Stated period. **Filters.** Stated filters. **Grain.** Stated population. **Unit.** Count of units. **Required fields.** quantity.

Interpretation. The volume half of every revenue movement. Revenue up with units flat is a price or mix story. **Common failure.** Counting order lines instead of units. A line carrying a quantity of three is one line and three units. **Verification.** Units must be at least as large as the line count on a file with no zero-quantity lines.

Owned by. Chapter 5, Section 5.7.

B.2.10 Number of orders

Question. How many purchase occasions occurred? **Formula.** Count of distinct order_id.

Numerator. Distinct order_id values. **Denominator.** None — it is a count. **Window.** Stated period. **Filters.** Stated filters. **Grain.** Stated population. **Unit.** Count of orders. **Required fields.** order_id.

Interpretation. The denominator of average order value and the volume term in the customer-value identity. **Common failure.** Using the row count of a line-grain file. A distinct count is not a row count; nunique on order_id counts orders and size counts order lines. **Verification.** Distinct orders must be no greater than the line count, and must equal the row count of any order-grain roll-up of the same file.

Owned by. Chapter 4, Section 4.8.

B.2.11 Number of customers

Question. How many distinct people bought? **Formula.** Count of distinct customer_id.

Numerator. Distinct customer_id values. **Denominator.** None — it is a count. **Window.** Stated period. **Filters.** Stated filters; lines carrying no customer_id cannot be counted and must be resolved, not dropped silently. **Grain.** Stated population. **Unit.** Count of customers. **Required fields.** customer_id.

Interpretation. The denominator of every customer-grain metric in Section B.3. **Common failure.** Letting a group-by drop rows whose customer_id is missing. pandas excludes missing group keys by default, so groups vanish without comment and the denominator shrinks unannounced. **Verification.** The customer count of a roll-up must equal the count of distinct non-null customer_id values in the lines it was built from.

Owned by. Chapter 4, Section 4.8.

B.2.12 Revenue growth rate

Question. How much did revenue change against its comparison period? **Formula.** (Current-period revenue − comparison-period revenue) / comparison-period revenue.

Numerator. Change in revenue between the two periods. **Denominator.** Comparison-period revenue. A comparison period of zero leaves the growth rate undefined; report the absolute change instead. **Window.** Two periods of equal length, both named; the comparison period is part of the metric, not context for it. **Filters.** Identical filters in both periods, including store roster. **Grain.** Stated population. **Unit.** Percentage. **Required fields.** line_revenue, order_date.

Interpretation. Meaningless without its comparison period. Year over year, quarter over quarter, and against plan are three different numbers. **Common failure.** Comparing periods whose store rosters differ, so an opening or closing is read as growth. Chapter 2's resort stores grew 40 percent partly because the prior year held a partial season. **Verification.** Confirm both periods carry the same length, the same filters, and a stated store roster. State the comparison period in the same sentence as the number.

Owned by. Chapter 2, Section 2.7.

B.2.13 Variance from target or baseline

Question. How far is the outcome from the number it was measured against? **Formula.** Actual − baseline for a dollar variance; (actual − baseline) / baseline for a percentage variance.

Numerator. Actual less the declared baseline. **Denominator.** The baseline, for the percentage form only. A baseline of zero leaves the percentage variance undefined; report the dollar variance instead. **Window.** The baseline's own period. **Filters.** The baseline's own filters. **Grain.** Stated population. **Unit.** Dollars for the difference; percentage for the ratio; never both under one label. **Required fields.** The actual, and a baseline declared before results were inspected.

Interpretation. The variance is only as meaningful as the baseline. A target set after the results are known is a description, not a benchmark. **Common failure.** Reporting a percentage variance and a dollar variance under one word and letting the reader assume which. State the unit. **Verification.** Confirm the baseline was declared in advance and record where. Recompute both forms and check they tell the same story.

Owned by. Chapter 2, Section 2.7.

B.2.14 Percentage change and percentage-point change

Question. Did the quantity move by a share of itself, or by an amount on its own scale? **Formula.** Percentage change = (new − old) / old. Percentage-point change = new − old, when both are already percentages.

Numerator. For percentage change, the difference; for percentage-point change, the difference itself is the answer. **Denominator.** For percentage change, the old value — an old value of zero leaves it undefined, and the change should be reported in levels; for percentage-point change, none. **Window.** Two stated periods. **Filters.** Identical in both periods. **Grain.** Stated population. **Unit.** Percentage, or percentage points — these are different units and are never interchangeable. **Required fields.** The two values being compared.

Interpretation. A repeat-order share moving from 20 to 24 percent rose by 4 percentage points and by 20 percent. Both are true; they are not the same claim. **Common failure.** Reporting the larger of the two because it is more persuasive. Chapter 13 names this as framing: presenting a quantity in the form that produces the intended reaction where the alternative framing is equally true. **Verification.** Say the unit aloud. If the quantity being compared is itself a percentage, the difference is in points and the ratio is in percent.

Owned by. Chapter 13, Section 13.15.

B.2.15 Worked example

Take the practice file's twenty order lines. Gross sales — quantity times catalog unit price, before discount — come to $1,350.00. Three lines carry a discount, giving up $38.00, so net revenue is $1,312.00. That is the figure every other example in this appendix reconciles to.

In the practice file unit cost is exactly half of catalog unit price, so cost is $675.00 and contribution is $637.00, a contribution margin of 48.6%. The discount lines are where the margin story lives: a full-price line contributes

exactly 50 percent, a line discounted 20 percent contributes 37.5 percent, and a line discounted 25 percent contributes 33.3 percent — because the discount comes off revenue and not off cost. Note that this is the practice file's own margin and not the 46 percent Chapter 13 computes for the suburban and resort stores; a margin belongs to the population it was computed on.

B.3 Basket and Customer Value Metrics

This family answers the question the expansion case turns on: what is a purchase occasion worth, what is a customer worth, and are those the same question. They are not.

B.3.1 Average order value (AOV)

Question. What does a typical purchase occasion come to? **Formula.** Sum of line_revenue / count of distinct order_id. In Tableau, SUM([line_revenue]) / COUNTD([order_id]).

Numerator. Net revenue on the included lines. **Denominator.** Distinct orders on the same included lines. **Window.** Stated period, both ends inclusive. **Filters.** Stated channel, store, metro, and customer filters, applied identically to numerator and denominator. **Grain.** One order. It is an order-grain mean: it is ordinarily computed from a line-grain file, but as revenue over distinct orders rather than as an average of line values. **Unit.** Dollars. **Required fields.** line_revenue, order_id.

Interpretation. Basket size in dollars. It knows nothing about frequency, so a store can lead on average order value and trail badly on revenue per customer. **Common failure.** Computing AVG(line_revenue) on a line-grain file. There is no error message and no visual difference between the correct chart and the incorrect one; Chapter 12 reports $50.41 for the line-grain average against $108.00 for the true chain figure. **Verification.** Recompute as total revenue over distinct orders and confirm the two agree. On the practice file: $1,312.00 over 12 orders is $109.33.

Common variants. Some organizations compute AOV net of returns, and some at gross sales. Both are defensible; state which.

Owned by. Chapter 5, Section 5.7.

B.3.2 Average of group average order values (do not use)

Question. None. This entry exists so the quantity can be recognized and rejected. **Formula.** Unweighted mean of each group's AOV.

Numerator. Sum of the group AOVs. **Denominator.** Count of groups. **Window.** As stated. **Filters.** As stated. **Grain.** One group, equally weighted. **Unit.** Dollars. **Required fields.** Group-level AOV.

Interpretation. Reconciles to no total. A group of 40 orders and a group of 40,000 receive equal votes. **Common failure.** Reporting it as the overall figure. On the practice file the metro averages give $129.71 against a true pooled $109.33; Chapter 5's miniature gives $105.00 against a true $82.50. **Verification.** The pooled figure and the weighted mean must agree exactly. If a third number appears, it is this one.

Owned by. Chapter 5, Section 5.3.

B.3.3 Units per order (basket size)

Question. How many items are in a typical basket? **Formula.** Sum of quantity / count of distinct order_id.

Numerator. Units on the included lines. **Denominator.** Distinct orders on the same lines. **Window.** Stated period. **Filters.** Stated filters. **Grain.** One order. **Unit.** Units per order. **Required fields.** quantity, order_id.

Interpretation. The volume factor of the average-order-value identity. Chapter 5's designed finding is that the suburban gap is basket size and mix, not unit price. **Common failure.** Dividing by lines rather than orders, which understates basket size wherever orders carry more than one line. **Verification.** Must be at least 1 on a file with no zero-quantity orders. Multiply by revenue per unit and confirm the product is average order value exactly.

Owned by. Chapter 5, Section 5.7.

B.3.4 Revenue per unit (realized price per unit)

Question. What did an item actually sell for, after discount and mix? **Formula.** Sum of line_revenue / sum of quantity.

Numerator. Net revenue on the included lines. **Denominator.** Units on the same lines. **Window.** Stated period. **Filters.** Stated filters. **Grain.** One unit. The denominator is units — not lines and not orders. **Unit.** Dollars per unit. **Required fields.** line_revenue, quantity.

Interpretation. The price factor of the average-order-value identity. It moves with category mix, discount depth, and promotional participation, not only with list price. **Common failure.** Reading it as a list price. StyleCraft's catalog prices do not differ by store; realized revenue per unit does, because mix does. **Verification.** Must fall inside the catalog band once discounts are accounted for. Recompute at a common price set to separate mix from price.

Owned by. Chapter 5, Section 5.7.

B.3.5 Average order value decomposition

Question. Is a basket-size gap made of more items or dearer items? **Formula.** AOV = units per order × revenue per unit. Exactly, by construction: revenue/orders = units/orders × revenue/units.

Numerator. Not applicable — a multiplicative identity. **Denominator.** Not applicable. **Window.** Stated period, identical for both factors. **Filters.** Identical for both factors. **Grain.** Order for the composite; order and unit for the two factors. **Unit.** Ratio factors, not additive shares. **Required fields.** line_revenue, quantity, order_id.

Interpretation. The identity yields factors that multiply, not percentage shares that add. On the practice file a 2.82x AOV gap between suburban and urban stores factors into a 1.50x basket factor and a 1.88x price factor. **Common failure.** Writing that 'basket size accounts for sixty percent of the gap.' The identity supplies no such allocation; an additive attribution needs a logarithmic decomposition and a stated convention. **Verification.** The two factors must multiply back to the composite exactly. Assert it rather than eyeballing it.

Owned by. Chapter 5, Section 5.7.

B.3.6 Revenue per customer

Question. What is one customer worth over the window? **Formula.** Sum of line_revenue / count of distinct customer_id.

Numerator. Net revenue attributed to the counted customers. **Denominator.** Distinct customers in the stated population. **Window.** A stated window. Without one this is a lifetime total wearing a period label. **Filters.** Stated filters; state whether non-purchasers are inside the denominator. **Grain.** One customer. **Unit.** Dollars per customer. **Required fields.** line_revenue, customer_id.

Interpretation. The customer-value counterpart of average order value. Revenue per customer equals AOV times orders per customer, which is where a weak repeat rate taxes a strong basket. **Common failure.** Interchanging it with average order value. Per order and per customer differ by frequency, and the expansion case must be argued on the customer figure. **Verification.** Confirm the identity: AOV × orders per customer = revenue per customer. On the practice file, $109.33 × 1.5 = $164.00.

Common variants. Computed over purchasers only, or over the whole customer file including non-purchasers. The two differ materially; name the base.

Owned by. Chapter 5, Section 5.7.

B.3.7 Orders per customer (purchase frequency)

Question. How often does a customer buy in the window? **Formula.** Count of distinct order_id / count of distinct customer_id.

Numerator. Distinct orders. **Denominator.** Distinct customers in the stated population. **Window.** A stated window; the figure is not comparable across windows of different length. **Filters.** Stated filters. **Grain.** One customer. **Unit.** Orders per customer. **Required fields.** order_id, customer_id.

Interpretation. The frequency factor between average order value and revenue per customer. **Common failure.** Comparing frequency across customers observed for different lengths of time. A customer who signed up

nine weeks ago cannot have accumulated two years of orders; see exposure months. **Verification.** Must be at least 1 among purchasers. Recompute as the mean of the customer-level order counts and confirm agreement.

Owned by. Chapter 6, Section 6.5.

B.3.8 Frequency (RFM)

Question. How many purchase occasions has this customer had in the window? **Formula.** Count of distinct order_id for the customer in the window.

Numerator. Distinct orders placed by the customer. **Denominator.** None — it is a count. **Window.** The declared analysis window, open at the start and closed at the end, measured to a declared analysis date. **Filters.** Customers with at least one order in the window. **Grain.** One customer. **Unit.** Count of orders. **Required fields.** order_id, customer_id, order_date.

Interpretation. Counts orders, not lines: an order of five lines is one act of purchasing, not five. **Common failure.** Counting at line grain, which inflates frequency for multi-line baskets and quietly rewards the suburban shopper the analysis is trying to measure. **Verification.** The sum of customer frequencies must equal the distinct order count of the window.

Owned by. Chapter 6, Section 6.5.

B.3.9 Monetary value (RFM)

Question. How much has this customer spent in the window? **Formula.** Sum of line_revenue for the customer in the window.

Numerator. Net revenue on the customer's lines. **Denominator.** None — it is a total. **Window.** The declared analysis window. **Filters.** Customers with at least one order in the window. **Grain.** One customer. **Unit.** Dollars. **Required fields.** line_revenue, customer_id, order_date.

Interpretation. Strongly right-skewed at StyleCraft, and bounded by how long the customer has been observed. **Common failure.** Comparing monetary totals across customers of very different tenure, which restates the tenure question rather than answering the performance one. **Verification.** The sum of customer monetary values must equal the window's certified revenue total.

Common variants. A common variant uses average order value in place of total revenue. Total rewards frequent modest baskets; average rewards rare large ones. The choice is declared, not defaulted.

Owned by. Chapter 6, Section 6.5.

B.3.10 Recency

Question. How long has it been since this customer last bought? **Formula.** Analysis date − the customer's most recent order_date, in days.

Numerator. Elapsed days since the most recent purchase. **Denominator.** None — it is a duration. **Window.** Measured back from a declared analysis date; undefined for a customer who has never purchased. **Filters.** Purchasing population; every order_date must fall on or before the analysis date. **Grain.** One customer. **Unit.** Days. **Required fields.** order_date, customer_id, a declared analysis date.

Interpretation. Lower is better, which is why the recency score reverses the raw days. **Common failure.** A negative recency, which is a date defect rather than a customer behavior and means an order is dated after the analysis date. **Verification.** Assert that every recency is at or above zero, and that the minimum is consistent with the snapshot the features were built at.

Owned by. Chapter 6, Section 6.5.

B.3.11 Exposure months

Question. How long has this customer actually been observable inside the window? **Formula.** (Analysis date − the later of signup_date and the window start) in days, divided by 30.44, floored at 1.0 month.

Numerator. Observed days inside the window. **Denominator.** 30.44 days per month, then floored at 1.0. **Window.** The declared analysis window. **Filters.** Purchasing population. **Grain.** One customer. **Unit.** Months. **Required fields.** signup_date, a declared analysis date and window start.

Interpretation. A denominator only. It exists so that a customer observed for nine days does not appear to have a stable monthly rate. **Common failure.** Flooring the day count rather than the months, which produces a minimum of 0.986 rather than 1.0 and lets a near-zero denominator through. **Verification.** Assert the minimum is exactly 1.0.

Owned by. Chapter 6, Section 6.6.

B.3.12 Orders per month

Question. How often does this customer buy, adjusted for how long we have watched? **Formula.** frequency / exposure_months.

Numerator. Distinct orders in the window. **Denominator.** Exposure months. **Window.** The declared analysis window. **Filters.** Purchasing population. **Grain.** One customer. **Unit.** Orders per month. **Required fields.** frequency, exposure_months.

Interpretation. Reduces the mechanical effect of tenure on frequency. It does not eliminate it from a specification that still contains the raw totals. **Common failure.** Treating it as independent of frequency. It is frequency divided by a constant-per-customer denominator, and standardization does not remove that redundancy. **Verification.** Least stable for customers observed shortest; report the exposure distribution beside it.

Owned by. Chapter 6, Section 6.6.

B.3.13 Revenue per exposure-month

Question. What does this customer spend per month of observed relationship? **Formula.** monetary / exposure_months.

Numerator. Net revenue in the window. **Denominator.** Exposure months. **Window.** The declared analysis window. **Filters.** Purchasing population. **Grain.** One customer. **Unit.** Dollars per month. **Required fields.** monetary, exposure_months.

Interpretation. The measure on which a short-tenured segment stops looking weak. A segment can trail badly on lifetime revenue and lead on revenue per month of exposure. **Common failure.** Aggregating it to segment level by averaging the customer-level ratios, which gives a $40 customer and a $1,000 customer equal votes. Pool the numerators and denominators instead, or say plainly that the figure is a mean of ratios. **Verification.** State which aggregation was used, and use the same one for every ratio in the same exhibit.

Owned by. Chapter 6, Section 6.6.

B.3.14 Discount share

Question. How much of this customer's spending came on discounted merchandise? **Formula.** Revenue on lines with discount_pct greater than 0 / monetary.

Numerator. Net revenue on the customer's discounted lines. **Denominator.** The customer's total net revenue in the window. **Window.** The declared analysis window. **Filters.** Discounted lines are those with discount_pct greater than 0; the roll-up is at line grain, because an order mixing a discounted dress with a full-price scarf is one order and two different facts. **Grain.** One customer. **Unit.** Proportion between 0 and 1. **Required fields.** line_revenue, discount_pct, customer_id.

Interpretation. A margin proxy at customer level. A customer with no discounted lines has a share of exactly 0, which is data rather than missingness. **Common failure.** Computing it at order grain, which assigns a whole mixed basket to one side. **Verification.** Bounded in [0, 1] with a small tolerance, since shares summed from rounded line revenue can land a fraction above one in floating point.

Owned by. Chapter 6, Section 6.6.

B.3.15 Store share

Question. How much of this customer's buying happens in physical stores? **Formula.** Distinct orders with channel of Store / frequency.

Numerator. The customer's distinct store-channel orders. **Denominator.** The customer's distinct orders in the window. **Window.** The declared analysis window. **Filters.** Channel is read at order grain; the build asserts one

channel per order. **Grain.** One customer. **Unit.** Proportion between 0 and 1. **Required fields.** channel, order_id, customer_id.

Interpretation. Separates the store-only occasion shopper from the digital-first core. **Common failure.** Taking the first channel of a multi-channel order, which manufactures a consistent order out of an inconsistent one. **Verification.** Bounded in [0, 1]. Assert that no order carries two channels before computing it.

Owned by. Chapter 6, Section 6.6.

B.3.16 Occasionwear share

Question. How much of this population's buying is occasionwear? **Formula.** Occasionwear units / total units for the unit-weighted form; occasionwear revenue / total revenue for the revenue-weighted form.

Numerator. Units, or revenue, on lines flagged is_occasionwear. **Denominator.** Total units, or total revenue, on the same lines. **Window.** Stated period. **Filters.** Requires a validated many-to-one join to products; an unmatched line has no flag and must not be silently counted as not occasionwear. **Grain.** Stated population — customer, metro, or store. **Unit.** Proportion. **Required fields.** quantity or line_revenue, is_occasionwear, product_id.

Interpretation. The mix term behind the suburban price factor. On the practice file the suburban store share is 0.500 against 0.000 urban. **Common failure.** Quoting 'occasionwear is 60 percent of suburban sales' without saying whether sales means units or revenue. The two differ whenever occasionwear prices differ from the rest. **Verification.** Bounded in [0, 1]. Confirm the join produced no unmatched lines before trusting the denominator.

Owned by. Chapter 5, Section 5.7.

B.3.17 Revenue share by decile

Question. How concentrated is revenue across customers? **Formula.** Decile revenue / total revenue in the window, over customers ranked by revenue and cut into ten equal-count groups.

Numerator. Revenue of the customers in the decile. **Denominator.** Total revenue in the window, across the ranked population. **Window.** A stated window — the interval is open at the start and closed at the end; concentration is window-dependent and the window belongs in the sentence. **Filters.** Customers with a resolved customer_id who were active in the window; ties broken deterministically so the cut is reproducible. **Grain.** One decile of customers. **Unit.** Proportion of revenue. **Required fields.** line_revenue, customer_id, order_date.

Interpretation. Decile 1 is the highest-revenue tenth. A top-decile share well above ten percent is the ordinary finding, not a surprising one. **Common failure.** Quoting a top-decile share from a short window. Measured concentration is inflated by observation error over short windows, so a 30-day decile share is not comparable with a trailing-twelve-month one. **Verification.** The deciles must partition: customer counts sum to the ranked population and decile revenues sum to the window total.

Common variants. The Pareto shorthand — the top 20 percent account for X percent — is the same construction at a different cut. The true split is rarely the folkloric 80/20, and it varies meaningfully by category and window (Schmittlein et al., 1993).

Owned by. Chapter 5, Section 5.9.

B.3.18 Worked example

The practice file's twelve orders and $1,312.00 give an average order value of $109.33. Computing the same name at line grain gives $65.60, and averaging the three store-type figures gives $129.71. All three are arithmetically correct; only the first answers the question. Revenue per customer is a different quantity again: $1,312.00 over eight customers is $164.00, which is the average order value times 1.5 orders per customer.

Table B.5 decomposes the store-type gap. Suburban orders average $220.00 against $78.00 urban, a factor of 2.82. That factors exactly into a 1.50 basket factor and a 1.88 realized-price factor, and the two multiply back to the composite. The price factor is a mix effect rather than a pricing one: the unit-weighted occasionwear share — occasionwear units over total units, not over revenue — is 50.0% of suburban units against 0.0% of urban units, and StyleCraft's catalog prices do not differ by store.

Table B.5

The average-order-value decomposition on the practice file

Store type	Orders	Revenue	Units	AOV	Units per order	Revenue per unit	Product of the two factors
Suburban	2	440.00	6	220.00	3.00	73.33	220.00
Urban	3	234.00	6	78.00	2.00	39.00	78.00

The last column is the verification the identity entry asks for: units per order times revenue per unit returns average order value exactly, on every row. What the identity does not supply is an allocation. It is correct to say the gap factors into basket and price terms; it is not correct to say that basket size accounts for some percentage of it.

B.4 Retention and Customer Relationship Metrics

Retention measures are where a metric name most often outlives the definition it was given, because "came back" admits so many defensible operational readings. Every entry in this family therefore carries an unusually explicit window and eligibility rule, and the family's first entry carries two named variants rather than one silent choice.

B.4.1 Repeat purchase rate

Question. What share of customers came back? **Formula.** Customers with two or more orders in the window / customers eligible in the window.

Numerator. Customers with two or more completed orders in the window. **Denominator.** Customers with at least one order in the window. **Window.** Trailing twelve months to a declared analysis date, bounded at both ends. A one-sided filter makes the reported rate depend on the day the notebook happens to run. **Filters.** All channels. The headline definition applies no channel filter. **Grain.** One customer. **Unit.** Proportion. **Required fields.** order_id, customer_id, order_date.

Interpretation. StyleCraft's headline retention measure and the baseline of the expansion review, owned by the CRM manager. Customers acquired late in the window have had little time to repurchase, a mechanical depressant a same-age cohort comparison is designed to remove. **Common failure.** Quoting the name without the definition. Three defensible definitions of this phrase circulate at StyleCraft and produce 0.60, 0.43, and 0.20 on the same eighteen orders. **Verification.** Bounded in [0, 1]. Hand-count the numerator and denominator on a small file: on Chapter 3's preview, three repeaters over seven active customers is 0.43.

Common variants. Lifetime: two or more orders ever, over all customers with at least one completed order in the available history — 0.60 on the preview, and unresponsive to current behavior. Store-only: the same window restricted to the store channel in both numerator and denominator — 0.20 on the preview, and the right measure for the economics of the physical footprint. Report the store version beside the headline rather than instead of it.

Owned by. Chapter 3, Section 3.10.3.

B.4.2 Repeat-order share

Question. What share of orders were placed by customers who had bought before? **Formula.** Orders placed after the customer's first purchase date / distinct orders.

Numerator. Orders whose order_date is later than the customer's first purchase date. **Denominator.** Distinct orders in the population. **Window.** Stated period; the first purchase date is computed on full history, before any date filter, so that trimming a partial month does not manufacture first purchases. **Filters.** The go/no-go dashboard excludes the reserved ONLINE store, because the comparison is across physical store types. **Grain.** One order. **Unit.** Percentage. **Required fields.** order_id, customer_id, order_date.

Interpretation. An order-grain measure. On the practice file it is 4 of 12, or 33.3 percent, against a customer-grain repeat purchase rate of 37.5 percent on the same file. **Common failure.** Labelling it 'returning-customer share'. It is not the share of customers who returned, and the obvious name invites a customer-level reading the metric does not support. Because order_date carries a date rather than a timestamp, every order placed on a customer's first purchase date is classified as a first-day order. **Verification.** Bounded in [0, 1]. Confirm the chain

figure is computed on the pooled denominator; Chapter 12 reports a chain share of 47 percent against an unweighted mean of the four store-type rates of 34 percent.

Owned by. Chapter 12, Section 12.13.2.

B.4.3 Churn rate (descriptive)

Question. What share of a stated customer population went quiet? **Formula.** Customers inactive by a stated rule / the stated customer population.

Numerator. Customers meeting the stated inactivity rule. **Denominator.** The stated eligible population. **Window.** A stated observation window. **Filters.** A stated eligibility rule. **Grain.** One customer. **Unit.** Proportion. **Required fields.** order_id, customer_id, order_date.

Interpretation. Descriptive bookkeeping. Churn is not naturally observed in a non-contractual business; it is operationalized, and different defensible rules produce different populations. **Common failure.** Presenting a descriptive churn rate as though it measured departure. It measures silence under one rule. **Verification.** Bounded in [0, 1]. State the rule and the window in the same sentence as the number.

Owned by. Chapter 5, Section 5.7.

B.4.4 Churn label

Question. Which customers are to be treated as having churned, for modeling? **Formula.** churned = 1 for an eligible customer with zero completed purchase orders in the outcome window.

Numerator. Customers with no outcome-window purchase order. **Denominator.** None — it is a per-customer flag, not a rate. **Window.** A six-month outcome window following a declared snapshot date; the feature window ends at the snapshot and both ends of each window are enforced. **Filters.** Both conditions: signup_date on or before the snapshot, and at least one purchase inside the feature window, so that every labeled customer has a history to be scored on. **Grain.** One customer. **Unit.** Binary flag. **Required fields.** order_id, customer_id, order_date, signup_date.

Interpretation. A measurement decision before it is a modeling one. Shortening the outcome window inflates the base rate with mid-cadence occasion shoppers; lengthening it purifies the label but treats customers too late to save. **Common failure.** Defining churn as zero net revenue rather than zero purchase activity. An order can net to zero after a discount and a returned line, so the two events are not the same and a revenue-based label mislabels them. **Verification.** Reconcile against the shipped answer key. Where activity and revenue definitions disagree, the disagreement is a finding about the revenue column rather than a bug in the label.

Common variants. A 90-day or 12-month outcome window, and activity redefined to include an app open or an email click, are each defensible and each produce a different base rate and a different treated population.

Owned by. Chapter 9, Section 9.3.

B.4.5 Cohort retention rate (period-active)

Question. Of the customers who started together, how many were active at a given age? **Formula.** Customers in the cohort active in period p / cohort size.

Numerator. Distinct customers in the cohort with at least one order in that calendar period. **Denominator.** Cohort size — the customers whose first order fell in the cohort's own period. **Window.** Periods measured from each cohort's own start, with an observability mask so that unreached periods stay missing rather than becoming zero. **Filters.** Deduplicated on customer and period, so a customer with three orders in a month counts once. **Grain.** One cohort in one period since its start. **Unit.** Percentage. **Required fields.** order_id, customer_id, order_date.

Interpretation. A period-active rate can rise as well as fall, because a customer who skipped a month can return. **Common failure.** Reading a blank cell as zero. Elapsed but inactive is 0.0; not yet reached is missing. A table that does not distinguish the two cannot be read down a column. **Verification.** Period zero must be 1.0 for every cohort; cohort sizes must sum to the distinct customer count; no cell may exceed 1.0.

Common variants. A survival retention rate holds the share retained continuously and is non-increasing by construction. A table that does not say which form it is cannot be read.

Owned by. Chapter 5, Section 5.8.

B.4.6 Cohort size

Question. How many customers started together? **Formula.** Count of customers whose first order fell in the cohort period.

Numerator. Distinct customers with a first order in the period. **Denominator.** None — it is a count. **Window.** The cohort's own defining period. **Filters.** Stated acquisition filters. **Grain.** One cohort. **Unit.** Count of customers. **Required fields.** customer_id, order_date.

Interpretation. Read down the left edge of every cohort table. A triumphant retention story over shrinking cohorts is two findings, not one. **Common failure.** Presenting retention percentages without the sizes that produced them. **Verification.** Cohort sizes must sum to the distinct customer count of the population.

Owned by. Chapter 5, Section 5.8.

B.4.7 Tenure

Question. How long has this person been a customer? **Formula.** Analysis date − signup_date, in days.

Numerator. Elapsed days since signup. **Denominator.** None — it is a duration. **Window.** Measured to a declared analysis or snapshot date. **Filters.** None beyond the stated population. **Grain.** One customer. **Unit.** Days. **Required fields.** signup_date, a declared analysis date.

Interpretation. The variable behind most apparent differences between new and established markets. Chapter 9's fairness screen exists because a new store's customers look unresponsive largely because they are new. **Common failure.** Reading a tenure artifact as a value verdict. A lifecycle position is not a quality judgment. **Verification.** Tenure must be at least recency for every customer; a violation indicates a broken join rather than an unusual customer.

Owned by. Chapter 6, Section 6.6.

B.4.8 Time to second purchase

Question. How long does it take a new customer to come back? **Formula.** Days between a customer's first and second order_date.

Numerator. Elapsed days between the two orders. **Denominator.** None — it is a duration. **Window.** Measured from each customer's own first order, to a declared analysis date. **Filters.** Customers with at least two orders; customers with only one are censored, not missing, and must be reported as such. **Grain.** One customer. **Unit.** Days. **Required fields.** order_id, customer_id, order_date.

Interpretation. Averaging only over customers who did come back understates the true wait, because the customers who never returned are the slowest of all. **Common failure.** Reporting a mean over repeaters as though it described the base. State the share of customers with no second purchase in the same sentence. **Verification.** Report the censored share beside the statistic.

Owned by. Chapter 5, Section 5.7.

B.4.9 Worked example

The practice file's eight customers placed twelve orders. Three customers placed two or more, so the repeat purchase rate over the file's own window is 0.375. The repeat-order share is a different quantity computed on the same twelve orders: 4 of them were placed after the customer's first purchase date, so the share is 0.333. The two numbers are close, they are not the same metric, and the difference is the denominator — customers in the first, orders in the second.

Section B.1.3 supplies the larger demonstration. On Chapter 3's eighteen-order preview, three defensible definitions of the first metric give 0.60, 0.43, and 0.20. Each is correct; each answers a different question; and the middle one is the definition StyleCraft adopted as its headline, with the store version reported beside it rather than instead of it.

B.5 Acquisition, Media, and Campaign Metrics

Acquisition and media metrics arrive from platforms that computed them under their own conventions, which is what makes them dangerous. The denominators are set by someone else, the attribution window is a configuration value with a budget attached, and the revenue in the numerator is attributed rather than incremental.

B.5.1 Impressions

Question. How many times was the advertising served? **Formula.** Count of served impressions, as reported by the platform.

Numerator. Impressions recorded by the serving platform. **Denominator.** None — it is a count. **Window.** The campaign window. **Filters.** One campaign, or a stated roll-up of campaigns. **Grain.** One campaign. **Unit.** Count. **Required fields.** impressions.

Interpretation. A delivery count, not a count of people. Impressions are not reach. **Common failure.** Using impressions as the denominator of a conversion rate, which mixes a served-advertising count with a customer count and produces a number that is not a funnel stage. **Verification.** Must be at least the click count on the same campaign.

Owned by. Chapter 1, Section 1.10.

B.5.2 Clicks

Question. How many times did someone click through? **Formula.** Count of recorded clicks.

Numerator. Clicks recorded by the serving platform. **Denominator.** None — it is a count. **Window.** The campaign window. **Filters.** One campaign, or a stated roll-up. **Grain.** One campaign. **Unit.** Count. **Required fields.** clicks.

Interpretation. Serves as numerator of click-through rate and denominator of post-click conversion rate, which is why its definition has to travel with both. **Common failure.** Treating clicks as unique people. Unless the platform de-duplicates, a click count is an event count. **Verification.** Must be no greater than impressions. Under a one-conversion-per-click convention clicks must also be at least the conversion count; otherwise compare the two only after confirming that clicks and conversions use compatible event definitions, since one click can carry credit for several conversion events.

Owned by. Chapter 1, Section 1.10.

B.5.3 Click-through rate (CTR)

Question. How well did the creative earn attention from the advertising served? **Formula.** clicks / impressions.

Numerator. Clicks. **Denominator.** Impressions. **Window.** The campaign window. **Filters.** One campaign, or a stated pooled roll-up; a campaign with zero impressions has an undefined rate, not an infinite one. **Grain.** One campaign. **Unit.** Proportion. **Required fields.** clicks, impressions.

Interpretation. A creative and targeting measure. It says nothing about what happened after the click. **Common failure.** Averaging campaign click-through rates to get a portfolio figure. Recompute from pooled totals: on Chapter 1's four campaigns the pooled rate is 0.0500. **Verification.** Bounded in [0, 1]. A rate above 1 means clicks exceed impressions somewhere and the numerator and denominator have been reversed or the join has duplicated.

Owned by. Chapter 1, Section 1.10.

B.5.4 Post-click conversion rate

Question. Of the units that clicked or entered through the campaign, what share converted under the declared counting convention? **Formula.** conversions / clicks.

Numerator. Attributed conversion events on the campaign, counted on the same basis as the denominator. **Denominator.** Click events on the same campaign, counted on the same basis as the numerator. **Window.** The campaign window, plus the platform's conversion window. **Filters.** One campaign; the column is named for its denominator so that the definition travels with the number. Numerator and denominator must count the same unit:

unique clickers against unique converters, sessions against converted sessions, or click events against conversion events under a stated one-conversion-per-click convention. **Grain.** One campaign. **Unit.** Proportion. **Required fields.** conversions, clicks.

Interpretation. One of at least four quantities the phrase conversion rate can denote. Conversions may be divided by clicks, sessions, visitors, leads, or recipients, and the results differ. **Common failure.** Reporting it as 'conversion rate' with no denominator named, or averaging the four campaign rates instead of pooling. Chapter 1's campaigns give 0.1161 pooled against 0.1179 averaged. **Verification.** Bounded in [0, 1] only when numerator and denominator count the same unit on the same basis; a value above 1 means they do not, most often because one converter was credited with several conversions against a single click. Confirm the conversion window is the same across any campaigns being compared.

Common variants. Conversion over impressions, over sessions, and over recipients are all in circulation. They are different metrics, not different estimates of one.

Owned by. Chapter 1, Section 1.10.

B.5.5 Cost per click (CPC)

Question. What did each click cost? **Formula.** spend / clicks.

Numerator. Media spend on the campaign. **Denominator.** Clicks on the same campaign. **Window.** The campaign window. **Filters.** One campaign; a campaign with zero clicks has an undefined cost per click. **Grain.** One campaign. **Unit.** Dollars per click. **Required fields.** spend, clicks.

Interpretation. An auction-efficiency measure. Low cost per click with low conversion buys traffic, not customers. **Common failure.** Letting a zero denominator produce an infinite value. A rate over nothing is undefined, not infinite, and a table full of inf is a denominator problem wearing a printing error. **Verification.** Recompute from pooled totals when aggregating; never average campaign-level rates.

Owned by. Chapter 5, Section 5.7.

B.5.6 Cost per thousand impressions (CPM)

Question. What did it cost to reach the audience a thousand times? **Formula.** spend / impressions × 1,000.

Numerator. Media spend. **Denominator.** Impressions, scaled to thousands. **Window.** The campaign window. **Filters.** One campaign. **Grain.** One campaign. **Unit.** Dollars per thousand impressions. **Required fields.** spend, impressions.

Interpretation. A buying-price measure that says nothing about response. Chapter 1's Clearance campaign runs at $139.71 against New Arrivals at $34.62. **Common failure.** Comparing cost per thousand across channels whose impression definitions differ. A video impression and a display impression are not the same event. **Verification.** Confirm the factor of one thousand is applied once. Recompute from pooled totals when aggregating.

Owned by. Appendix B; the book computes spend and impressions but does not name this ratio.

B.5.7 Cost per conversion

Question. What did each conversion cost in advertising? **Formula.** ad_spend / conversions.

Numerator. Advertising spend on the campaign. **Denominator.** Conversions attributed to the same campaign. **Window.** The campaign window plus the conversion window. **Filters.** One campaign; advertising spend only, excluding creative production and agency fees. A campaign with zero conversions has an undefined cost per conversion and is reported separately. **Grain.** One campaign. **Unit.** Dollars per conversion. **Required fields.** ad_spend, conversions.

Interpretation. A campaign-level efficiency measure. It is not customer acquisition cost, because a conversion is not necessarily a new customer. **Common failure.** Reading it as acquisition cost. Repeat purchases convert too. **Verification.** Recompute from pooled totals when aggregating. Confirm the spend column excludes non-media cost, or rename it.

Owned by. Chapter 1, Exercise 1.4.

B.5.8 Customer acquisition cost (CAC)

Question. What did it cost to acquire one new customer? **Formula.** Acquisition spend / customers acquired.

Numerator. Acquisition spend, on a stated cost basis. **Denominator.** Customers acquired, on a stated definition of acquired. **Window.** A stated window, with the window and the definition of acquired doing most of the work. **Filters.** Stated channel and campaign attribution rules. **Grain.** One acquired customer, reported for a cohort. **Unit.** Dollars per customer. **Required fields.** spend, customer_id, signup_date or first order_date.

Interpretation. The most reliably contested marketing metric, because two defensible cost bases produce two very different numbers and each answers a different question. **Common failure.** Comparing a campaign-attributed figure against a fully loaded one. Marketing's number — campaign spend over attributed new customers — is usually and sometimes substantially smaller than finance's, which divides total sales and marketing expense by every new customer, attributed or not. **Verification.** State the cost basis, the attribution rule, and the definition of acquired in the same place as the number. If the organization cannot state all three, that is the finding.

Common variants. Campaign cost per acquisition against fully loaded customer acquisition cost. Neither is wrong; they are different metrics and must not be compared.

Owned by. Chapter 5, Section 5.7; the dispute is set out in Chapter 3, Section 3.12.2.

B.5.9 Return on ad spend (ROAS)

Question. How much revenue came back for each advertising dollar? **Formula.** revenue / ad_spend.

Numerator. Revenue attributed to the campaign. **Denominator.** Advertising spend on the campaign, excluding creative production and agency fees. **Window.** The campaign window plus the attribution window. **Filters.** One campaign; a campaign with zero advertising spend has an undefined return on ad spend and is reported separately rather than assigned an infinite value. **Grain.** One campaign. **Unit.** Ratio. **Required fields.** revenue, ad_spend.

Interpretation. Attributed, not incremental. Return on ad spend answers what revenue was credited to the campaign, not what revenue the campaign caused. **Common failure.** Two distinct failures. First, ignoring margin: Chapter 1's Clearance campaign returns 2.46 on revenue and destroys $5,288 of contribution. Second, reading it causally: an attributed figure becomes an incremental one only through an experiment or a designed holdout. **Verification.** Recompute from pooled totals when aggregating; the four campaigns pool to 5.79 against an average of 12.33. Then compute net contribution after advertising spend beside it.

Common variants. Incremental return on ad spend, measured against a randomized or geographic holdout, is a different and stronger quantity. Use the word incremental only when the evidence supports it.

Owned by. Chapter 1, Section 1.10.

B.5.10 Attributed orders

Question. How many orders were credited to a campaign? **Formula.** Count of distinct order_id among lines carrying the campaign_id.

Numerator. Distinct orders carrying the campaign identifier. **Denominator.** None — it is a count. **Window.** Stated period. **Filters.** Lines with a non-null campaign_id; a null campaign_id may mean unattributed or may mean attribution failed, and the two look identical in the file. **Grain.** One campaign. **Unit.** Count of orders. **Required fields.** order_id, campaign_id.

Interpretation. The product of an attribution rule, not an observation. An attributed order may be credited on a view rather than a click. **Common failure.** Dropping null campaign_id rows before computing campaign performance, which assumes every null is a legitimate non-attribution. **Verification.** Attributed orders summed across campaigns must not exceed the distinct order count of the file. An order carrying two campaign identifiers would be counted once under each while that reconciliation still passed.

Owned by. Chapter 5, Section 5.8.

B.5.11 Attribution (conversion) window

Question. How long after an interaction may a conversion still be credited to it? **Formula.** A declared number of days following a click, view, or engaged view.

Numerator. Not applicable — it is a parameter of other metrics. **Denominator.** Not applicable. **Window.** It is the window. **Filters.** Set per conversion action and per platform. **Grain.** Applies to every attributed metric downstream. **Unit.** Days. **Required fields.** A platform setting, recorded alongside every attributed figure.

Interpretation. A configuration value with a budget attached. Shortening it reduces the conversions the account records; the customers do not change. **Common failure.** Comparing channels measured on different windows. Every platform ships its own defaults, so this happens easily and the comparison is not close to fair. **Verification.** Record the window beside every attributed figure and refuse to compare channels whose windows differ.

Owned by. Chapter 3, Section 3.12.3.

B.5.12 Worked example

Chapter 1's four campaigns are the guide's canonical media example, and Table B.6 computes the family on them. Read the return on ad spend column first and the Loyalty Offer is the obvious winner at 28.00. Read the last column and the Clearance campaign, which returned $2.46 for every advertising dollar, destroyed $5,288 of contribution — because its margin rate is 0.18 and return on ad spend does not know that.

Table B.6

The acquisition and media family, computed on Chapter 1's four campaigns

Campaign	CTR	Post-click conv.	CPC	CPM	Cost per conv.	ROAS	Net contribution after ad spend
New Arrivals	0.0550	0.1084	0.63	34.62	5.81	15.50	13,545
Loyalty Offer	0.0620	0.1654	0.47	29.27	2.86	28.00	9,552
Clearance	0.0500	0.1147	2.79	139.71	24.36	2.46	-5,288
Welcome Series	0.0400	0.0833	1.73	69.33	20.80	3.37	3,550

The table also shows why rates are recomputed rather than averaged. Pooling the four campaigns gives a click-through rate of 0.0500 and a post-click conversion rate of 0.1161; averaging the four campaign figures gives 0.1179 for the second, which reconciles to no total. The distortion is far worse on return on ad spend, where the pooled figure is 5.79 and the average of the four campaign ratios is 12.33 — more than twice as large, because the average gives the tiny, efficient Loyalty Offer the same weight as the large, inefficient Clearance campaign.

B.6 CRM and Engagement Metrics

Engagement metrics share one problem and it is the denominator. Sent, delivered, and unique-opened bases all circulate under the same metric names, and they produce different numbers from identical behavior.

B.6.1 Delivery rate

Question. What share of messages reached an inbox? **Formula.** delivered / sent.

Numerator. Messages accepted by the receiving server. **Denominator.** Messages sent. **Window.** The send, or a stated period of sends. **Filters.** One send, or a stated roll-up. **Grain.** One send. **Unit.** Percentage. **Required fields.** sent, delivered.

Interpretation. The denominator every downstream engagement rate should probably use. **Common failure.** Ignoring it, and then computing open rate on sends without saying so. **Verification.** Delivery rate and bounce rate sum to 1 only once every message in the send has reached a final delivered-or-bounced status and the denominator excludes suppressed, pending, and otherwise unresolved records. State that condition, or state the share still unresolved.

Owned by. Appendix B; the book names deliverability but catalogs no formula.

B.6.2 Bounce rate

Question. What share of messages failed to reach an inbox? **Formula.** bounced / sent.

Numerator. Messages rejected, hard or soft. **Denominator.** Messages sent. **Window.** The send. **Filters.** One send; state whether hard and soft bounces are combined. **Grain.** One send. **Unit.** Percentage. **Required fields.** sent, bounced.

Interpretation. A list-health measure. A rising bounce rate is a data-quality finding before it is a marketing one. **Common failure.** Combining hard and soft bounces silently, which hides a list-decay signal inside a transient one. **Verification.** Bounce rate and delivery rate sum to 1 only under the condition stated in the delivery-rate entry: every message resolved, and no suppressed or pending records in the denominator.

Owned by. Appendix B.

B.6.3 Open rate

Question. What share of recipients opened the message? **Formula.** Unique opens / delivered. The headline definition counts unique opens; total opens is a different numerator and a different metric.

Numerator. Unique opens — one per recipient who opened at least once. **Denominator.** Delivered messages, or sent messages — these are different denominators. **Window.** The send, plus a stated attribution period for late opens. **Filters.** One send, or a stated roll-up. **Grain.** One send. **Unit.** Percentage. **Required fields.** unique_opens, delivered, sent.

Interpretation. A weak behavioral measure. Technical privacy protections pre-fetch images on the recipient's behalf, so an open may record no human attention at all. **Common failure.** Circulating three denominators under one name. On the worked example, 2,688 opens is 28.0 percent of delivered and 26.9 percent of sent — both correct, and not the same metric. **Verification.** Bounded in [0, 1] for the unique-open form only. A total-open rate is not bounded by 1, because one recipient can open repeatedly, and it should be reported as opens per delivered message rather than as a percentage. State the base — sent or delivered — in the same sentence, and never compare an open rate across a period in which the platform changed its open measurement.

Common variants. Unique opens over delivered is the headline. Unique opens over sent is defensible and systematically smaller. Total opens over delivered is a different quantity altogether: it counts events rather than recipients, has no upper bound of 1, and must not be labeled a rate alongside the other two.

Owned by. Chapter 5, Section 5.7.

B.6.4 Click rate

Question. What share of recipients clicked? **Formula.** Unique clicks / delivered. The headline definition counts unique clicks; total clicks is a different numerator and is not bounded by 1.

Numerator. Unique clicks — one per recipient who clicked at least once. **Denominator.** Delivered, or sent. **Window.** The send, plus a stated attribution period. **Filters.** One send, or a stated roll-up. **Grain.** One send. **Unit.** Percentage. **Required fields.** unique_clicks, delivered, sent.

Interpretation. A stronger behavioral signal than open rate, because it is harder to trigger without intent. **Common failure.** Comparing a click rate computed on delivered against one computed on sent. On the worked example, 384 clicks is 4.00 percent of delivered and 3.84 percent of sent. **Verification.** Bounded in [0, 1] for the unique-click form only; a total-click count per delivered message has no upper bound of 1. Do not assume the click rate cannot exceed the open rate: a click can be recorded when the open-tracking pixel was blocked or never fired, so the ordering is an expectation rather than a guarantee, and a reversal is a tracking finding rather than an arithmetic error.

Owned by. Chapter 5, Section 5.7.

B.6.5 Click-to-open rate (CTOR)

Question. Of the recipients who opened, how many went on to click? **Formula.** Unique recipients with both a recorded open and a recorded click / unique opens.

Numerator. Unique recipients who both opened and clicked. This is not the same as all unique clickers, because a click can be recorded when no open was. **Denominator.** Unique opens. **Window.** The send. **Filters.** One send; numerator and denominator drawn from the same recipient population. **Grain.** One send. **Unit.** Percentage.

Required fields. unique_opens, unique_clicks, and the recipient-level join that identifies which clickers also opened.

Interpretation. A content and offer measure, holding the subject line constant. On the worked example every clicker also registered an open, so 384 over 2,688 is 14.3 percent. **Common failure.** Taking all unique clickers as the numerator. Because the click-rate entry establishes that a click can be recorded when the open pixel never fired, that numerator is not a subset of the denominator and the result can exceed 100 percent. **Verification.** Bounded in [0, 1] by construction, because the numerator is a subset of the denominator. Under the all-clickers variant the bound does not hold, and a value above 1 is a tracking finding — open measurement is incomplete — rather than an arithmetic error. Because opens are inflated by automatic pre-fetching, this rate is deflated by the same mechanism.

Common variants. Some platforms report all unique clickers over unique opens, which is easier to compute and is not bounded by 1. Where that form is used, say so and treat any figure above 100 percent as evidence about the tracking rather than about the creative.

Owned by. Appendix B.

B.6.6 Unsubscribe rate

Question. What did this message cost in permission? **Formula.** Unsubscribes / delivered.

Numerator. Unsubscribe events attributed to the send. **Denominator.** Delivered messages. **Window.** The send, plus a stated attribution period. **Filters.** One send. **Grain.** One send. **Unit.** Percentage. **Required fields.** unsubscribes, delivered.

Interpretation. A guardrail rather than a performance measure. A campaign that wins on click rate and loses permission has not won. **Common failure.** Omitting it from a campaign readout because it is small. The cost is cumulative and irreversible. **Verification.** Bounded in [0, 1]. Where the field is not instrumented, say so: a preregistered guardrail that cannot be calculated is a design defect, not a minor gap.

Owned by. Chapter 11, Table 11.3, where it is named as an instrumentation obligation.

B.6.7 Opt-in rate

Question. What share of customers have consented to be contacted on this channel? **Formula.** Customers with the channel's consent flag set / customers in the stated population.

Numerator. Customers with email_opt_in true, or with sms_opt_in true — one channel at a time, never combined. **Denominator.** Customers in the stated population. **Window.** A point in time, as of a stated date. **Filters.** Stated population; one named channel. **Grain.** One customer, reported for a population. **Unit.** Percentage. **Required fields.** email_opt_in or sms_opt_in, customer_id.

Interpretation. A permission measure. Because a Boolean is stored as 1 and 0, its mean is a legitimate and useful quantity. **Common failure.** Folding app_user into this rate. Installing an application is not consent to be contacted, and the two belong in separate entries; see the app-user share below. A second failure is reading opt-in as interest, when it may measure the aggressiveness of the signup flow in force when the customer joined. **Verification.** Bounded in [0, 1]. Confirm the flag is genuinely Boolean and that missing values have not been coerced to false. Report one channel per figure.

Owned by. Chapter 3, Section 3.4.1.

B.6.8 App-user share

Question. What share of customers have the mobile application, as of today? **Formula.** Customers with app_user true / customers in the stated population.

Numerator. Customers with app_user true. **Denominator.** Customers in the stated population. **Window.** A point in time, as of a stated date. This is a stock, not a flow: it is not an adoption or activation rate over a window. **Filters.** Stated population. **Grain.** One customer, reported for a population. **Unit.** Percentage. **Required fields.** app_user, customer_id.

Interpretation. A reachability measure and a strong behavioral correlate — Chapter 7 spends a whole section on why that correlation is not an effect. **Common failure.** Treating it as consent, or as an adoption rate. app_user records installation at a point in time; it carries no permission and no date, so a change between two extracts cannot

be attributed to a period. **Verification.** Bounded in [0, 1]. State the as-of date, since the field is overwritten rather than versioned.

Owned by. Appendix B; the guide uses app_user as a feature and a holdout but catalogs no share.

B.6.9 Worked example

One send of 10,000 messages: 400 bounced, so 9,600 were delivered. 2,688 unique opens and 384 unique clicks followed — every clicker in this send also registered an open, which is what lets the click-to-open rate be read as a share — with 48 unsubscribes. Table B.7 computes the family twice, once on each base, and the point of the table is the gap between the two columns.

Table B.7

One email send, with every rate computed on both circulating bases

Metric	On delivered	On sent	Which base the entry recommends
Delivery rate	—	96.0%	Sent, always
Bounce rate	—	4.0%	Sent, always
Open rate	28.0%	26.9%	Delivered, stated in the same sentence
Click rate	4.0%	3.8%	Delivered, stated in the same sentence
Click-to-open rate	14.3%	—	Opens; the numerator counts only clickers who also opened
Unsubscribe rate	0.50%	—	Delivered

Neither column is wrong. An open rate of 28.0 percent and one of 26.9 percent describe identical behavior, and a team that reports one while its benchmark uses the other will conclude something false about a difference that does not exist. This is the Chapter 3 dispute in its smallest form, and the remedy is the same: name the base beside the number.

B.7 Funnel Metrics

A funnel is a sequence of stages counting the same unit, in one window, under an event-order requirement. Where any of those three conditions fails, the arithmetic still runs and the result is not a funnel.

B.7.1 Stage count

Question. How many units reached this stage? **Formula.** Count of units meeting the stage's membership rule.

Numerator. Units at the stage. **Denominator.** None — it is a count. **Window.** A single stated window, identical across every stage. **Filters.** A stated membership rule per stage, and a stated event-order requirement. **Grain.** The funnel's unit — session, visitor, customer, or lead. One unit for the whole funnel. **Unit.** Count. **Required fields.** Whatever the stage rules require, plus a stage timestamp.

Interpretation. Every stage of a funnel must count the same kind of thing. Sessions, clicks, and orders are three different units, not three stages. **Common failure.** Building a funnel whose stages count different units and calling the last figure a conversion rate. **Verification.** Each stage's count must be no greater than the stage above it, once the event-order requirement is enforced.

Owned by. Chapter 5, Section 5.8.

B.7.2 Stage-to-stage conversion rate

Question. Of the units that reached this stage, how many reached the next? **Formula.** Count at this stage / count at the previous stage.

Numerator. Units at this stage. **Denominator.** Units at the immediately preceding stage. **Window.** One window, shared by both stages. **Filters.** Shared membership rules; both stages must count the same unit. **Grain.** The funnel's unit. **Unit.** Percentage. **Required fields.** Stage counts.

Interpretation. Localizes the loss. On the worked funnel, carts to orders converts at 40.0 percent while sessions to product views converts at 30.0 percent. **Common failure.** Quoting a step rate and a cumulative rate under one

word. A reader told one while imagining the other is misinformed by true numbers. **Verification.** Bounded in [0, 1]. The product of the step rates must equal the cumulative rate at the bottom: 0.300 × 0.250 × 0.400 = 0.030.

Owned by. Chapter 5, Section 5.8.

B.7.3 Cumulative (overall) conversion rate

Question. Of everyone who entered, how many finished? **Formula.** Count at this stage / count at the top of the funnel.

Numerator. Units at this stage. **Denominator.** Units at the funnel's top stage. **Window.** One window. **Filters.** Shared membership rules. **Grain.** The funnel's unit. **Unit.** Percentage. **Required fields.** Stage counts.

Interpretation. The end-to-end figure. On the worked funnel it is 3.0 percent from session to order. **Common failure.** Computing it across stages drawn from different populations or periods, or dividing orders by impressions and calling the result a funnel conversion rate. Impressions are advertising delivery, not funnel entrants. **Verification.** Must equal the product of the step rates. If it does not, a stage is counting a different unit.

Owned by. Chapter 5, Section 5.8.

B.7.4 Drop-off rate

Question. What share of units were lost at this step? **Formula.** 1 − stage-to-stage conversion rate.

Numerator. Units that reached the previous stage and not this one. **Denominator.** Units at the previous stage. **Window.** One window. **Filters.** Shared membership rules. **Grain.** The funnel's unit. **Unit.** Percentage. **Required fields.** Stage counts.

Interpretation. The complement of the step rate, and often the more actionable framing. **Common failure.** Adding drop-off rates across steps. They are computed on different denominators and do not sum. **Verification.** Drop-off and step conversion must sum to 1 at every step.

Owned by. Appendix B; the book computes step rates but does not name the complement.

B.7.5 Worked example

A four-stage funnel counting sessions throughout: sessions 40,000, product views 12,000, carts 3,000, orders 1,200. Table B.8 gives the step and cumulative forms.

Table B.8

One funnel, read two ways

Stage	Count	Step conversion	Drop-off	Cumulative conversion
Sessions	40,000	—	—	100.0%
Product views	12,000	30.0%	70.0%	30.0%
Carts	3,000	25.0%	75.0%	7.5%
Orders	1,200	40.0%	60.0%	3.0%

The two rates describe the same funnel and answer different questions. Carts convert to orders at 40.0 percent, the strongest step in the funnel; sessions convert to orders at 3.0 percent, which is the number an executive means by conversion rate. The verification the entries require is that the step rates multiply to the cumulative rate: 0.300 × 0.250 × 0.400 = 0.030. If they do not, a stage is counting something other than sessions, and the exhibit is not a funnel.

B.8 Segmentation and Targeting Measures

Segmentation measures describe groups the analyst constructed, which means they carry a risk the other families do not: they can look like findings about customers when they are findings about the specification. Several entries below are therefore validity measures rather than business measures, and each states what it does and does not establish.

B.8.1 Segment size

Question. How many customers are in this segment? **Formula.** Count of customers assigned to the segment.

Numerator. Customers in the segment. **Denominator.** None — it is a count. **Window.** The window in which the segmenting features were built. **Filters.** The declared segmentation population — at StyleCraft, customers with at least one order in the window. **Grain.** One segment. **Unit.** Count of customers. **Required fields.** customer_id, the segment assignment.

Interpretation. Sizes are findings, not footnotes: size is the first row of the profile. **Common failure.** Treating size alone as substantiality. Whether a segment is worth serving is a threshold question about whether differentiated execution pays, not a headcount test. **Verification.** Segment sizes must sum to the population that was segmented.

Owned by. Chapter 6, Section 6.10.

B.8.2 Segment share of customers

Question. What portion of the base does this segment represent? **Formula.** Customers in the segment / customers in the segmented population.

Numerator. Customers in the segment. **Denominator.** Customers in the segmented population, which is not the same as the customer file. **Window.** The feature window. **Filters.** The declared segmentation population. **Grain.** One segment. **Unit.** Proportion. **Required fields.** customer_id, the segment assignment.

Interpretation. On the practice file, NYC Suburban holds 37.5 percent of customers and 45.8 percent of revenue. **Common failure.** Quoting the share against the whole customer file when the segmentation excluded non-purchasers. **Verification.** Shares must sum to 1.

Owned by. Chapter 6, Section 6.11.

B.8.3 Segment share of revenue

Question. What portion of revenue does this segment produce? **Formula.** Segment revenue / total revenue in the window.

Numerator. Revenue of the segment's customers. **Denominator.** Total revenue of the segmented population. **Window.** The feature window. **Filters.** The declared segmentation population. **Grain.** One segment. **Unit.** Proportion. **Required fields.** line_revenue, customer_id, the segment assignment.

Interpretation. Read beside the customer share. A segment holding a fifth of customers and a half of revenue is a different proposition from one holding both. **Common failure.** Reporting a segment revenue share that does not add back to the certified window total, which means customers were lost between the feature table and the summary. **Verification.** Shares must sum to 1, and segment revenues must sum to the certified window revenue.

Owned by. Chapter 6, Section 6.11.

B.8.4 RFM scores (R, F, M)

Question. Where does this customer rank on recency, frequency, and spending? **Formula.** Quintile scores from 1 to 5 on recency and monetary value, with recency reverse-scored; frequency scored in declared bands where the data will not support five equal-count bins.

Numerator. Rank position within the scored population. **Denominator.** Five bins, equal-count for quintiles and declared-threshold for bands. **Window.** The declared analysis window. **Filters.** The scored population; every customer with the same frequency must receive the same frequency score. **Grain.** One customer. **Unit.** Ordinal score, unitless. **Required fields.** recency_days, frequency, monetary.

Interpretation. Scores preserve rank, not level. The top monetary quintile exists even when spending has fallen for every customer in the base. **Common failure.** Forcing five equal-count frequency bins on a lumpy count. Where a large share of customers have exactly one order, quintiles either fail outright or split identical customers on row order, which carries no behavioral content. **Verification.** Assert that customers sharing a frequency share a score. Report the cut points beside the scores, since the scores are not comparable across periods or populations without them.

Common variants. Monetary value may be scored on total revenue or on average order value. Total rewards frequent modest baskets; average rewards rare large ones.

Owned by. Chapter 6, Section 6.5.

B.8.5 RFM grid region

Question. Which named behavioral region does this customer fall in? **Formula.** A published lookup from the (R, F) score pair to a named region.

Numerator. Not applicable — a lookup, not a ratio. **Denominator.** Not applicable. **Window.** The declared analysis window. **Filters.** The scored population. **Grain.** One customer. **Unit.** Categorical label. **Required fields.** R and F scores.

Interpretation. The grid sees exactly three behaviors and draws its regions on two of them, so the largest-spending customer can land in an at-risk region. **Common failure.** Publishing a grid with gaps or overlaps, which lets two analysts apply the same rules and reach different assignments. **Verification.** Assert the grid is exhaustive and mutually exclusive over all twenty-five score pairs.

Owned by. Chapter 6, Section 6.5.

B.8.6 Silhouette score

Question. How cleanly separated is a clustering solution? **Formula.** For each customer, the comparison of average distance to its own cluster against average distance to the nearest other cluster, averaged over customers.

Numerator. Not applicable — a scaled comparison. **Denominator.** Not applicable. **Window.** The feature window. **Filters.** The standardized clustering inputs. **Grain.** One customer, then averaged across the solution. **Unit.** Unitless score between -1 and +1. **Required fields.** The standardized feature matrix and the cluster assignment.

Interpretation. Near +1 means snugly placed, near 0 means on a boundary, negative means arguably misassigned. **Common failure.** Obeying it. The silhouette often crowns the coarsest cut, because the coarsest cut is the cleanest, even when finer structure is real and strategically necessary. **Verification.** Report it beside the inertia curve and the business constraint on the number of treatments. It grades geometry only: a high score says the partition is clean, not that its groups are marketing segments.

Owned by. Chapter 6, Section 6.9.

B.8.7 Adjusted Rand index (ARI)

Question. Do two clustering solutions agree about which customers belong together? **Formula.** Over all pairs of customers, how often two solutions agree about whether the pair belongs together, corrected for the agreement expected by chance.

Numerator. Not applicable — a chance-corrected pairwise agreement. **Denominator.** Not applicable. **Window.** The feature window. **Filters.** The customers both solutions saw. **Grain.** One pair of customers, summarized per pair of partitions. **Unit.** Index, unitless; 1.0 for identical partitions, near 0 for chance-level agreement, and negative for less agreement than chance, with a lower bound of −0.5. **Required fields.** Two cluster assignments over the same customers.

Interpretation. Invariant to relabeling, which is why it replaces cross-tabulating dominant cells. Used for three distinct questions: stability across seeds, sensitivity to the feature set, and recovery of a designed answer key. **Common failure.** Reading reproducibility as meaning. High agreement establishes that the partition is reproducible; it does not establish that the groups are marketing segments. **Verification.** A solution compared with itself must return exactly 1.0; assert it as a control before trusting any other value.

Owned by. Chapter 6, Section 6.10.

B.8.8 Inertia (within-cluster sum of squares)

Question. How tightly do customers sit around their assigned cluster centers? **Formula.** Total of squared distances from customers to their assigned centroids.

Numerator. Not applicable — a sum of squared distances. **Denominator.** Not applicable. **Window.** The feature window. **Filters.** The standardized clustering inputs. **Grain.** One clustering solution at a given number of clusters. **Unit.** Squared standardized distance, unitless. **Required fields.** The standardized feature matrix and the fitted model.

Interpretation. Its level is uninformative; its shape across candidate cluster counts is the signal. It reaches zero at one cluster per customer. **Common failure.** Citing low inertia as validation. It is a fit statistic, not evidence that the partition means anything. **Verification.** Expect it to fall and flatten. A rise at a higher cluster count is a warning to inspect initialization and convergence rather than a finding.

Owned by. Chapter 6, Section 6.8.

B.8.9 Cluster centroid

Question. What does the typical member of this cluster look like? **Formula.** The mean of the cluster's members on every clustering feature, read in standardized units and then in native units.

Numerator. Sum of feature values within the cluster. **Denominator.** Customers in the cluster. **Window.** The feature window. **Filters.** The clustering features only; profiling attributes are read after assignment, never used as inputs. **Grain.** One cluster on one feature. **Unit.** Standard deviations, then the feature's own units. **Required fields.** The standardized feature matrix and the cluster assignment.

Interpretation. A centroid of -0.8 on recency means the cluster buys more recently than average by four fifths of a standard deviation. **Common failure.** Reading a centroid as a description of every member. Cluster means are means: a monetary centroid can be propped by a few large customers, so spreads and medians travel with it. **Verification.** Recompute the centroids as group means of the standardized features and confirm they match the fitted model's. A cluster whose mean contradicts its own median is not one story.

Owned by. Chapter 6, Section 6.10.

B.8.10 Worked example

The practice file's eight customers split into two home metros. Table B.9 gives the family's descriptive measures. The suburban group holds a smaller share of customers than of revenue, which is the signature the segmentation chapter is looking for and the reason both shares are reported rather than one.

Table B.9

Segment measures on the practice file's eight customers

Segment	Customers	Share of customers	Revenue	Share of revenue	Revenue per customer
NYC Suburban	3	37.5%	601.00	45.8%	200.33
NYC Urban	5	62.5%	711.00	54.2%	142.20

Both share columns sum to 1, which is the verification those entries require. What the table cannot show at this scale is the validity question the rest of the family exists to answer: whether a partition of eight customers into two groups is a finding about customers or an artifact of the split. That is what the adjusted Rand index, the stability check, and the holdout profiling in Section B.8 are for, and none of them can be run on eight rows.

B.9 Regression and Predictive-Model Evaluation Measures

Although this appendix catalogs marketing metrics, the guide relies repeatedly on measures of analytical performance, and mixing them with operating measures is how a fit statistic ends up on an executive scorecard. They are collected here and in Sections B.10 and B.11, clearly separated from the operating families above.

B.9.1 Pearson correlation coefficient (r)

Question. How strongly, and in which direction, do two quantities move together? **Formula.** Sample covariance of the two variables / the product of their two sample standard deviations.

Numerator. Sample covariance. **Denominator.** Product of the two sample standard deviations. **Window.** The window in which both variables were measured. **Filters.** Complete pairs. Software computes each cell of a correlation matrix over the rows complete for that pair, so one partly missing column produces a matrix whose cells describe different populations while presenting them as one exhibit. **Grain.** The grain at which both variables were

measured — state it, because r computed over stores and r computed over customers are different quantities. **Unit.** Unitless, between -1 and +1. **Required fields.** Two numeric variables measured for the same observational cases and at the same grain.

Interpretation. A linear, symmetric, pairwise association. It is not directional, so 'drivers' language does not follow from it. **Common failure.** Reading a correlation near 0.9 on individual behavior as a triumph. Near-perfect r usually means the two columns are the same fact wearing two names. **Verification.** Undefined when either variable has zero variance; software returns a missing value rather than zero. Always plot the scatter: identical statistics do not mean identical evidence.

Common variants. Population covariance divides by n and sample covariance by n − 1. Correlation is invariant to the choice; covariance is not.

Owned by. Chapter 7, Section 7.3.

B.9.2 R-squared

Question. How much of the outcome's variation does this model account for on this data? **Formula.** 1 − (residual variation / total variation), where total variation is the sum of squared deviations of the outcome around its own mean.

Numerator. Residual sum of squares, subtracted from 1 as a ratio. **Denominator.** Total sum of squares around the outcome's mean. **Window.** The model's own estimation sample. **Filters.** In-sample, with an intercept, on the same observations, for nested models. **Grain.** One model. **Unit.** Proportion under the stated conditions; it is not bounded below by zero out of sample. **Required fields.** Fitted values and actual outcomes.

Interpretation. A measure of fit to this data. It is not a percentage of predictions that were right, and it is not accuracy. **Common failure.** Reading a rising R-squared as a better model. Because it cannot decrease when a variable is added, a rise is not by itself evidence of anything. An R-squared above roughly 0.9 on individual customer behavior should be treated as a defect report rather than a result. **Verification.** Confirm the model carries an intercept, since the uncentered version computed without one is not comparable and is routinely larger. Out of sample the statistic can go negative, which is a sign that it is not a percentage of anything.

Owned by. Chapter 7, Section 7.7.

B.9.3 Regression coefficient

Question. How much does the outcome differ, per unit of this predictor, holding the model's other predictors fixed? **Formula.** The fitted slope from ordinary least squares; for a dummy, the difference against the reference category.

Numerator. Not applicable — a fitted parameter. **Denominator.** Not applicable. **Window.** The estimation window. **Filters.** The estimation sample; adding or removing a predictor changes every other coefficient's meaning. **Grain.** One predictor, in outcome units per predictor unit. **Unit.** Outcome units per predictor unit; for a dummy, outcome units relative to the reference. **Required fields.** The design matrix and the outcome.

Interpretation. A coefficient is not read until its units are said aloud. Ceteris paribus means other included things equal, and a perfectly adjusted difference is still an association. **Common failure.** Reading a dummy coefficient as a level. The wrong sentence is 'the model shows Suburban Occasion customers spend $X'; the coefficient is dollars relative to the reference category, not dollars absolutely. **Verification.** Check sign, magnitude, units, and reference category against the descriptive figures the same data produced. Report the coefficient and its interval from the same fit.

Owned by. Chapter 7, Section 7.8.

B.9.4 Confidence interval for a coefficient

Question. How precisely is this association estimated? **Formula.** The fitted coefficient plus and minus a multiple of its standard error, at a stated confidence level.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The estimation window. **Filters.** The estimation sample; robust and non-robust standard errors give different intervals from identical coefficients. **Grain.** One coefficient. **Unit.** Outcome units per predictor unit. **Required fields.** The coefficient and its standard error.

Interpretation. At large sample sizes nearly everything clears conventional significance, so the interval's width separates a great deal that the p-value separates hardly at all. **Common failure.** Quoting the coefficient from one fit and the p-value from another. Robust standard errors change the interval and leave the coefficient and the fitted values unchanged; mixing them answers two questions at once. **Verification.** The point estimate must lie inside its own interval. State the confidence level and the standard-error type.

Owned by. Chapter 7, Section 7.10.

B.9.5 p-value

Question. How surprising is this estimate, if the stated null hypothesis and model are correct? **Formula.** The probability, assuming the null and the model, of a test statistic at least as extreme as the one observed.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The estimation window. **Filters.** The stated null hypothesis, which for a coefficient is that it equals zero given the model's other variables. **Grain.** One hypothesis test. **Unit.** Probability between 0 and 1. **Required fields.** The test statistic and its reference distribution.

Interpretation. A measure of surprise under the null. It is not the probability that a hypothesis is true, statistical significance is not practical significance, and no single p-value should by itself determine a conclusion. **Common failure.** Reading small as important. On eight thousand customers, near-trivial associations clear the conventional threshold easily. **Verification.** Report it beside the effect size and the interval, never alone.

Owned by. Chapter 7, Section 7.10.

B.9.6 Mean absolute error (MAE)

Question. By how much does a numeric prediction typically miss? **Formula.** The mean of the absolute differences between actual and predicted values.

Numerator. Sum of absolute errors. **Denominator.** Number of evaluated cases. **Window.** The outcome window of the evaluated cases. **Filters.** Evaluated out of sample; on the test set for the final grade, and on training folds during development. **Grain.** One prediction, averaged over the evaluated population. **Unit.** The target's own units — dollars for a spend target. **Required fields.** Actual and predicted values.

Interpretation. The typical miss, with every error weighted equally. On the worked example the model misses by $14.00 against a mean baseline that misses by $36.88. **Common failure.** Reporting a distance metric alone for a ranking decision, which quietly substitutes a question the analyst can answer for the one the business asked. **Verification.** Lower is better, so a value far below the baselines is the suspicious result and far above is the broken one. Compare against the strongest applicable baseline on the same folds.

Owned by. Chapter 8, Section 8.6.

B.9.7 Root mean squared error (RMSE)

Question. By how much does a numeric prediction miss, weighting large misses more heavily? **Formula.** The square root of the mean squared difference between actual and predicted values.

Numerator. Sum of squared errors, square-rooted after averaging. **Denominator.** Number of evaluated cases. **Window.** The outcome window. **Filters.** Evaluated out of sample. **Grain.** One prediction, averaged over the evaluated population. **Unit.** The target's own units. **Required fields.** Actual and predicted values.

Interpretation. The squaring is the policy: it weights a single $300 miss like nine separate $100 misses. RMSE materially above MAE announces that error is concentrated in a few large misses. On the worked example, $17.36 against an MAE of $14.00. **Common failure.** Switching to RMSE after losing on the declared metric. A candidate that wins on RMSE after losing on the declared capture and MAE has not won; the goalposts have moved. **Verification.** RMSE is always at least MAE on the same predictions. For nested ordinary-least-squares models fitted on the same training observations, training RMSE cannot rise as terms are added, because least squares minimizes exactly this quantity; the guarantee does not extend to regularized or otherwise constrained models, and training MAE carries no corresponding guarantee under ordinary least squares.

Owned by. Chapter 8, Section 8.6.

B.9.8 Mean squared error (MSE)

Question. By how much does a numeric prediction miss, in squared units? **Formula.** The mean of the squared differences between actual and predicted values; the square of the RMSE on the same predictions.

Numerator. Sum of squared errors. **Denominator.** Number of evaluated cases. **Window.** The outcome window. **Filters.** Evaluated out of sample, on the same cases as any metric it is compared with. **Grain.** One prediction, averaged over the evaluated population. **Unit.** The square of the target's units — squared dollars for a spend target, which is why the guide reports RMSE instead. **Required fields.** Actual and predicted values.

Interpretation. Cataloged because the syllabus names it among the session outcomes, not because the guide reports it. It ranks candidate models identically to RMSE, since the square root is monotone. **Common failure.** Quoting it in a deliverable. Squared dollars are not a quantity any stakeholder can price, and the same information in readable units is one square root away. **Verification.** MSE must equal the square of the RMSE on the same predictions. Where an assignment asks for MSE, report RMSE beside it.

Owned by. Appendix B; the syllabus asks for it and no chapter defines it.

B.9.9 Baseline score

Question. What does the simplest defensible rule achieve under identical conditions? **Formula.** The mean baseline predicts the training mean for everyone; the majority-class baseline predicts the most common class; the last-value baseline predicts that the previous window repeats.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The candidate's own evaluation window. **Filters.** Exactly the candidate's conditions: same folds, same test set, same metric. The mean must be computed from training labels, never test labels. **Grain.** Matches the candidate. **Unit.** Matches the candidate's metric. **Required fields.** Training labels, and for the last-value rule the prior window's outcome.

Interpretation. The last-value rule is not a strawman; it is the incumbent, and the sorted spreadsheet costs nothing. **Common failure.** Choosing the baseline after the results, which allows a baseline to be chosen to lose. **Verification.** Declare the baseline before fitting. Computing the mean baseline from test labels hands it a summary of the answers.

Owned by. Chapter 8, Section 8.7.

B.9.10 Margin over baseline

Question. What did the model add over the rule it must beat? **Formula.** A signed difference from baseline, stated in the direction in which improvement is positive. For loss metrics such as MAE and RMSE, baseline score − candidate score; for gain metrics such as capture, candidate score − baseline score. Computed on the same folds and the same test set, and preferably as a paired per-fold difference reported with its spread and its count of folds won.

Numerator. The improvement on the shared metric, signed so that a positive value always means the candidate is better. **Denominator.** Not applicable. **Window.** Shared evaluation window. **Filters.** Identical population and folds. **Grain.** One candidate against one baseline. **Unit.** The metric's own units. **Required fields.** Both scores, computed identically.

Interpretation. A model's demonstrated value is its margin over the strongest applicable baseline, not its raw score. An MAE of $54 is good if the best naive rule scores $80 and embarrassing if it scores $55. **Common failure.** Subtracting in the wrong direction. Because half the metrics in this appendix improve downward and half improve upward, a single subtraction order cannot serve both, and a margin reported without its direction is unreadable. Separately: a margin smaller than the fold-to-fold spread of the metric it is measured on is not yet a finding. **Verification.** State which direction represents improvement in the same sentence as the number, and confirm the sign: a model that beats its baseline must produce a positive margin under either convention. Pair fold by fold: folds differ in difficulty, so subtracting the baseline within each fold cancels that and the spread of the differences shows whether the candidate reliably wins or merely averages well.

Owned by. Chapter 8, Section 8.9.

B.9.11 Capture at a selection share

Question. Of the outcome the population produced, how much did the customers we could afford to treat account for? **Formula.** Actual outcome of the top-k customers ranked by predicted score / actual outcome of the whole evaluated population, where k is the capacity divided by the population size.

Numerator. Actual outcome-window value of the top-k ranked customers. **Denominator.** Actual outcome-window value of the whole evaluated population. **Window.** The outcome window for the actuals; the snapshot for the ranking. **Filters.** The evaluated population; ties broken deterministically so the marginal name does not change between the analysis and the campaign build. **Grain.** One ranked list, reported at the population level. **Unit.** Percentage of outcome. **Required fields.** Predicted scores, actual outcomes, a customer key, and a declared capacity.

Interpretation. The decision metric when the program can treat only a fixed number of customers. Higher is better, which is the opposite direction from MAE and RMSE. **Common failure.** Reading a capture gap as revenue the program will generate. It says the customers the model selected turned out to spend more than the customers the spreadsheet selected; it says nothing about what either group would have spent untreated. **Verification.** A constant score carries no ranking information, so its capture is decided entirely by row order — which is why the mean baseline must break ties on a declared random draw. Compute the selection share from capacity rather than typing a proportion.

Owned by. Chapter 8, Section 8.6.

B.9.12 Selection share

Question. What fraction of the scorable population can the program actually treat? **Formula.** Program capacity / the population the model can honestly score.

Numerator. Funded treatment slots. **Denominator.** The eligible, scorable population. **Window.** As of the scoring run. **Filters.** The declared eligibility rule. **Grain.** One program, one population. **Unit.** Proportion. **Required fields.** A declared capacity and the eligible population count.

Interpretation. Capacity is a count, not a proportion: it does not grow if the eligible population turns out to be larger. The share is computed rather than typed. **Common failure.** Typing a proportion, which nobody re-derives when the population changes. **Verification.** Recompute whenever the eligibility rule changes. It is also the capture floor that random ordering achieves in expectation; any single realized random list scatters around it.

Owned by. Chapter 8, Section 8.6.

B.9.13 Worked example

Take eight customers whose actual six-month spend is known and a model that predicted it. The model's mean absolute error is $14.00 and its root mean squared error is $17.36. That RMSE sits above the MAE, which is the signal the entry describes: the error is not spread evenly but concentrated in a few larger misses.

Neither figure means anything alone. The mean baseline — predict $108.75 for everyone, computed from the training labels — posts a mean absolute error of $36.88. The model's margin is therefore $36.88 less $14.00, or $22.88, and that is the number the deliverable reports. Note the subtraction order. Mean absolute error improves downward, so its margin is baseline minus candidate; capture improves upward, so its margin is candidate minus baseline. The entry states both, because a margin reported without its direction cannot be read. A mean absolute error of $14 is a good result against a baseline of $37 and an embarrassing one against a baseline of $15.

B.10 Classification and Ranked-Targeting Measures

Classification measures are arithmetic summaries of four counts at one threshold, and almost every failure in this family comes from quoting one of them without the other three, without the threshold, or without the base rate. The family closes with the two ranked-targeting measures the retention program is actually budgeted from.

B.10.1 Base rate

Question. How common is the outcome we are trying to find? **Formula.** Positive cases / evaluated cases.

Numerator. Cases in the positive class. **Denominator.** All evaluated cases. **Window.** The outcome window. **Filters.** The declared eligible population. **Grain.** One case, reported for a population. **Unit.** Proportion. **Required fields.** The label.

Interpretation. The single most important number in a classification project and the one most reliably missing from celebrated results. It is a constructed number: a fact about the population under this label definition. **Common failure.** Reporting any classification metric without it. Accuracy, precision, and lift are uninterpretable in its absence. **Verification.** No classification result leaves the workspace without its base rate in the same exhibit. A drifting base rate silently re-prices the threshold even when the ordering holds.

Owned by. Chapter 9, Section 9.2.

B.10.2 Confusion-matrix counts (TP, FP, FN, TN)

Question. At this threshold, who did the model reach, miss, and correctly leave alone? **Formula.** Cross the actual class against the predicted class at a stated threshold; every evaluated case lands in exactly one of four cells.

Numerator. Not applicable — four counts. **Denominator.** Not applicable. **Window.** The outcome window. **Filters.** The evaluated set, at one stated threshold. **Grain.** One case; the four counts describe the evaluated set. **Unit.** Counts. **Required fields.** Predicted class at a stated threshold, and the actual label.

Interpretation. Every threshold-dependent classification metric is an arithmetic summary of these four counts, and changing the threshold changes the matrix. In campaign language: offer sent and customer was leaving; offer sent and customer was staying; no offer and customer left; no offer and customer stayed. **Common failure.** Pasting a matrix into a deck without re-adding it. A matrix that does not sum is a pasted matrix, and pasted matrices are how wrong numbers reach decks. **Verification.** Assert that the four counts sum to the evaluated set, then recompute precision and recall from the raw counts.

Owned by. Chapter 9, Section 9.6.

B.10.3 Accuracy

Question. What share of cases did the model classify correctly? **Formula.** (True positives + true negatives) / all evaluated cases.

Numerator. Correctly classified cases. **Denominator.** All evaluated cases. **Window.** The outcome window. **Filters.** The evaluated set, at one stated threshold. **Grain.** One case, reported for the evaluated set. **Unit.** Proportion. **Required fields.** Predicted class and actual label.

Interpretation. Under imbalance it is dominated by the model's performance on the class nobody is asking about. **Common failure.** Using it as a headline. On the worked ten-customer file, accuracy is 0.70 at both the 0.50 and the 0.30 threshold, while expected net value moves from $32 to $48 — accuracy has failed to detect a difference the program will pay for. **Verification.** Report it only beside the base rate and the majority-class baseline, and compute the margin between them.

Owned by. Chapter 9, Section 9.6.

B.10.4 Precision

Question. Of the customers we treated, how many were the real thing? **Formula.** True positives / (true positives + false positives).

Numerator. True positives. **Denominator.** All predicted positives — that is, everyone treated. **Window.** The outcome window. **Filters.** The evaluated set, at one stated threshold. **Grain.** One case, reported for the evaluated set. **Unit.** Proportion. **Required fields.** Predicted class and actual label.

Interpretation. The purity of the target list. On the worked file, precision of 0.50 means half the retention budget landed on customers who were staying anyway. **Common failure.** Quoting precision and recall achieved at different thresholds, or quoting either without its threshold. Precision is undefined when nothing is treated, and must be reported as undefined rather than as zero. **Verification.** Bounded in [0, 1]. Recompute from the raw counts. A low precision at a deliberately low threshold is by design, not a failure.

Owned by. Chapter 9, Section 9.7.

B.10.5 Recall (true positive rate)

Question. Of the customers who really were leaving, how many did we reach? **Formula.** True positives / (true positives + false negatives).

Numerator. True positives. **Denominator.** All actual positives. **Window.** The outcome window. **Filters.** The evaluated set, at one stated threshold. **Grain.** One case, reported for the evaluated set. **Unit.** Proportion. **Required fields.** Predicted class and actual label.

Interpretation. Coverage of the at-risk class. On the worked file, recall of 0.67 at the 0.50 threshold means a third of the departing customers walked out untreated. **Common failure.** Reading it without precision. The two move in opposite directions as the threshold falls, so either alone can be made to look good. **Verification.** Bounded in [0, 1]. Recall and the false negative rate must sum to 1.

Owned by. Chapter 9, Section 9.7.

B.10.6 F1 score

Question. How do precision and recall trade off, weighted equally? **Formula.** 2 × precision × recall / (precision + recall) — the harmonic mean.

Numerator. Twice the product of precision and recall. **Denominator.** Their sum. **Window.** The outcome window. **Filters.** The evaluated set, at one stated threshold. **Grain.** One evaluated set. **Unit.** Unitless, between 0 and 1. **Required fields.** Precision and recall.

Interpretation. Harmonic rather than arithmetic, so it punishes imbalance between the two: precision 0.9 with recall 0.1 posts an F1 near 0.18, not a flattering 0.5. **Common failure.** Optimizing it and calling the result optimal. It weights the two errors equally, and the entire threshold apparatus exists because the two errors are not equally priced. **Verification.** Bounded in [0, 1]. Use it only where costs are genuinely unpriced, and treat that as a temporary condition.

Owned by. Chapter 9, Section 9.7.

B.10.7 False positive rate and false negative rate

Question. How are the model's errors distributed across the two classes, and across groups? **Formula.** False positive rate = false positives / all actual negatives. False negative rate = false negatives / all actual positives.

Numerator. False positives, or false negatives. **Denominator.** All actual negatives, or all actual positives. **Window.** The outcome window. **Filters.** The evaluated set at the threshold the program will actually use; reported per group in a fairness screen. **Grain.** One evaluated set, or one group within it. **Unit.** Proportion. **Required fields.** Predicted class, actual label, and the grouping attribute.

Interpretation. The pair a fairness screen compares across groups, at the operating threshold rather than on average scores. **Common failure.** Computing them on small groups. A fairness appendix that reports a 0.00 false negative rate on a group of nine has produced noise with a decimal point. **Verification.** Report the group size beside every rate. This guide screens out groups below roughly fifty cases, which is a course convention rather than a statistical boundary: the precision of a group's error rates depends on how many positives and negatives it holds, not on its total size alone.

Owned by. Chapter 9, Section 9.12.10.

B.10.8 ROC AUC

Question. How well does the model order cases, independent of any threshold? **Formula.** The area under the curve traced by the true positive rate against the false positive rate across the full threshold sweep; equivalently, the probability that a randomly chosen positive outranks a randomly chosen negative, with ties counted as half.

Numerator. Not applicable — an area. **Denominator.** Not applicable. **Window.** The outcome window. **Filters.** Cross-validated on training folds during development; once on the sealed test set for the grade. **Grain.** One model on one population. **Unit.** Unitless, between 0 and 1; 0.5 is chance. **Required fields.** Continuous scores and the actual label.

Interpretation. A ranking measure and a poor sole criterion. A model with the highest overall AUC can be inferior in the top two deciles, inferior near the operating threshold, worse calibrated, and worth less money.

Common failure. Comparing a binary flag's AUC with a continuous score's. A flag has one nontrivial operating point and the two figures are not comparable. Separately: an AUC far above what the label definition and honest baselines would support is a leak until proven otherwise. **Verification.** Assert that a constant-score classifier returns 0.5. Multiplying every score by a constant leaves AUC unchanged while breaking every economic calculation built on those scores.

Owned by. Chapter 9, Section 9.8.

B.10.9 Calibration (reliability) check

Question. Do the model's predicted probabilities match observed event frequencies? **Formula.** Within score bins, compare the mean predicted probability against the observed positive rate; report the difference and the count in each bin.

Numerator. Observed positives within the bin; dividing by the bin's case count gives the observed positive rate. **Denominator.** Cases in the bin. **Window.** The outcome window. **Filters.** Out-of-fold predictions on training customers only. Calibration is repaired using training evidence, never by adjusting a model repeatedly against the test set. **Grain.** One score bin. **Unit.** Probability points. **Required fields.** Predicted probabilities, actual labels, and a declared binning.

Interpretation. Distinct from discrimination, and neither implies the other. Calibration is the property that entitles a threshold derived from expected costs to be applied to raw scores. **Common failure.** Applying an economically derived threshold to uncalibrated scores. That is not a small technical error; it prices a decision in a currency the model does not print. **Verification.** Print the count per bin, since a bin holding eleven customers cannot support a calibration verdict. Read the two bins straddling the operating threshold first: a model beautifully calibrated at 0.05 and off by fifteen points at 0.30 is miscalibrated where it matters.

Owned by. Chapter 9, Section 9.8.

B.10.10 Decision threshold

Question. Above what probability is treating a customer worth the cost? **Formula.** Cost of treating / expected benefit of treating a true positive. At StyleCraft, $12 / $40 = 0.30.

Numerator. The certain cost per treated customer. **Denominator.** The expected benefit per treated true positive, itself the save rate times the value of a save. **Window.** The program's planning horizon. **Filters.** Applied to calibrated probabilities. **Grain.** One policy. **Unit.** Probability. **Required fields.** A costed treatment, a save-rate estimate, and a value-of-a-save estimate.

Interpretation. Not a property of the model but a policy choice. The software default of 0.5 embodies the assumption that the two errors cost the same and inherits its authority from nothing else. **Common failure.** Accepting the default. Four conditions ride on the derived threshold: the scores must be calibrated, the cost inputs are estimates, the save rate is a causal claim that observational machinery cannot certify, and the arithmetic assumes every customer shares one benefit figure. **Verification.** Re-derive it from the stated costs and publish the sensitivity: at a $30 benefit the cut moves to 0.40, at $60 to 0.20. A threshold that swings widely under plausible inputs is telling the program that its economics are the binding uncertainty.

Owned by. Chapter 9, Section 9.9.

B.10.11 Treatment rate (treated share)

Question. What share of the population, or of a group, does this policy treat? **Formula.** Cases scoring at or above the threshold / cases in the population or group.

Numerator. Cases at or above the threshold. **Denominator.** Cases in the population or group. **Window.** As of the scoring run. **Filters.** The stated group, at the threshold the program will actually use. **Grain.** One population, or one group within it. **Unit.** Proportion. **Required fields.** Scores, the threshold, and the grouping attribute.

Interpretation. The economic threshold answers whom it is worth treating; the budget answers how many treatments are affordable, and nothing forces the two answers to agree. **Common failure.** Setting the treated-list size first — a round number, a platform tier limit, last year's list — and never reconciling it against the gains curve.

Verification. Report it beside the capacity. Where the threshold marks more customers than the budget covers, either cut by rank or take the gap to the meeting as a priced argument.

Owned by. Chapter 9, Section 9.9.

B.10.12 Expected net value

Question. What is this policy worth against its cost? **Formula.** True positives × benefit − treated × cost; per thousand customers, scaled by 1,000 / evaluated cases.

Numerator. Benefit earned on true positives, less cost spent on everyone treated. **Denominator.** For the per-thousand form, the evaluated case count. **Window.** The outcome window. **Filters.** One policy at one threshold. **Grain.** One policy, reported per population or per thousand customers. **Unit.** Dollars, or dollars per thousand customers. **Required fields.** The confusion-matrix counts, a costed treatment, and a benefit estimate.

Interpretation. On the worked file: treat nobody, $0; the model at 0.50, +$32; the model at 0.30, +$48; the ninety-day incumbent rule, -$8; treat everyone, exactly $0. The threshold, not the model, moved the program from $32 to $48. **Common failure.** Computing it from raw scores that were never checked for calibration. **Verification.** Per-thousand scaling keeps folds and populations of different sizes comparable. Confirm the blanket policy's arithmetic from the base rate alone as a check.

Owned by. Chapter 9, Section 9.9.

B.10.13 Lift by decile

Question. How much richer in the target class is each slice of the ranked file than a random one? **Formula.** The decile's positive rate / the base rate of the whole evaluated file.

Numerator. Positive rate within the decile. **Denominator.** The base rate of the evaluated file. **Window.** The outcome window. **Filters.** The evaluated file, ranked by score and cut into ten equal-count groups with ties broken deterministically. **Grain.** One decile of cases. **Unit.** Ratio, expressed as a multiple of the base rate. **Required fields.** Scores, labels, and a deterministic ranking.

Interpretation. A number with its baseline built in, which is why it survives in industry vocabulary. Lift should be strongest in the early deciles and weaken as the list deepens. **Common failure.** Hunting a non-monotone middle. Perfect monotonic decline is not guaranteed in a finite sample; a non-monotone middle is a finding about where the ordering blurs. What would be an error is a top decile whose lift sits near 1.0. **Verification.** Rank before cutting, so the ten groups are equal in size regardless of how many cases share a score. That is what makes lift comparable across deciles.

Owned by. Chapter 9, Section 9.10.

B.10.14 Cumulative gains

Question. Treating down to this depth of the list, what share of all positives do we reach? **Formula.** Running cumulative positives through this decile / total positives in the evaluated file.

Numerator. Cumulative count of actual positives through the decile. **Denominator.** Total positives in the evaluated file. **Window.** The outcome window. **Filters.** The evaluated file, score-ranked. **Grain.** One cumulative depth of the ranked file. **Unit.** Proportion of all positives. **Required fields.** Scores, labels, and a deterministic ranking.

Interpretation. Read against the diagonal that random targeting would trace. It reaches 1.0 at the whole file by definition. **Common failure.** Reading the curve correctly and then computing the program's return from raw scores nobody checked for calibration. **Verification.** Assert the last value is exactly 1.0. Plot the incumbent rule as a single point, not a curve: a fixed policy has one operating point.

Owned by. Chapter 9, Section 9.10.

B.10.15 Odds ratio

Question. By what factor do the odds of the outcome change per unit of this predictor? **Formula.** The exponentiated logistic coefficient.

Numerator. Not applicable — a multiplicative factor. **Denominator.** Not applicable. **Window.** The estimation window. **Filters.** The interpretive model, fitted without regularization and never used to treat anyone. **Grain.** One predictor. **Unit.** Multiplicative factor on the odds. **Required fields.** A fitted logistic coefficient.

Interpretation. Because the logistic curve is steepest at even odds, the same odds ratio moves a mid-risk customer's probability substantially and a low-risk customer's barely at all. Statements about percentage-point changes in probability are honest only with a stated starting point. **Common failure.** Reading these as causal. They are associations wearing precise multiplicative clothing: 'app users churn less' earns its verb while 'the app retains customers' does not. **Verification.** Report the interval on the odds-ratio scale. An interval straddling 1.0 is honest evidence of a weak association and should be reported as such rather than by quoting the point estimate alone.

Owned by. Chapter 9, Section 9.4.

B.10.16 Worked example

Ten scored customers, three of whom churned, give a base rate of 0.30. The retention program costs $12 per treated customer and a treated true churner is worth $40 in expectation, so the threshold is 12 divided by 40, or 0.30. Table B.10 prices the same model at that threshold and at the software default.

Table B.10

One model, two thresholds, and the measures that do and do not notice the difference

Measure	Threshold 0.50	Threshold 0.30
True positives	2	3
False positives	2	3
False negatives	1	0
True negatives	5	4
Customers treated	4	6
Accuracy	0.70	0.70
Precision	0.50	0.50
Recall	0.67	1.00
F1	0.57	0.67
Expected net value	$32	$48

Accuracy is 0.70 at both thresholds, and the majority-class baseline — predict that nobody churns — also scores 0.70. On this file accuracy is simultaneously unable to distinguish the two policies and unable to beat doing nothing, while expected net value moves from $32 to $48. The threshold, not the model, produced that gain. For comparison, the incumbent ninety-day lapse rule treats 4 customers, catches 1, and returns $-8.

Ranked down the file, the top two of the ten customers are both churners, so the first decile posts a lift of 3.33 against the base rate and the top two deciles capture 66.7% of all churners while treating 20% of the file. That is what the gains curve measures, and the budget stopping rule reads down it until the incremental capture in a decile is worth less than the decile costs to treat.

B.11 Forecast Evaluation Measures

Forecast evaluation has one measure that travels between series of different sizes and one catastrophic failure mode attached to it. The entries below keep the scale-free measure and its guard together, and add the signed measure the chapters use in code but define nowhere else. The scale-free measure and its failure at small denominators follow Hyndman and Koehler (2006).

B.11.1 Forecast error (signed)

Question. By how much, and in which direction, did the forecast miss? **Formula.** Actual − forecast, for each evaluated period.

Numerator. Actual less forecast. **Denominator.** None — it is a difference. **Window.** Each evaluated period inside the forecast horizon. **Filters.** Periods with both an actual and a forecast present. **Grain.** One period — for

StyleCraft, one day. **Unit.** The series' own units, in dollars. **Required fields.** Actual and forecast series aligned on the same index.

Interpretation. Positive means the forecast undershot. The sign convention must be stated, because the season-total miss is conventionally computed the other way round. **Common failure.** Reporting only absolute error, which hides a persistent one-directional miss. **Verification.** Confirm the sign convention in the same sentence as the number.

Owned by. Chapter 10, Section 10.7.

B.11.2 Forecast bias (mean signed error)

Question. Does this method systematically run high or low? **Formula.** The mean of the signed errors across the evaluated periods.

Numerator. Sum of signed errors. **Denominator.** Number of evaluated periods. **Window.** Rolling origins, or a sealed holdout. **Filters.** Evaluated periods only. **Grain.** One method on one series. **Unit.** The series' own units, signed. **Required fields.** Actual and forecast series.

Interpretation. Where a trending, expanding series announces whether a method is systematically low. Every candidate that cannot climb is biased low on an expanding series, which is a fact about the business rather than about the software. **Common failure.** Treating a run of same-signed errors as noise. Seven positive errors in a row is a persistent undershoot, not bad luck. **Verification.** Report it beside the absolute measures; mature demand-planning organizations track signed error as seriously as absolute error.

Owned by. Chapter 10; operationalized in code, and cataloged here because the chapter defines it nowhere else.

B.11.3 Mean absolute percentage error (MAPE)

Question. How large are the forecast's misses relative to the size of what actually happened? **Formula.** For each period, the absolute error divided by the actual, averaged across evaluated periods and multiplied by 100.

Numerator. Absolute error in each period. **Denominator.** That period's actual value. **Window.** The evaluated periods. **Filters.** Requires strictly positive actuals; disqualified for any series that touches or approaches zero. **Grain.** One period, then averaged. **Unit.** Percentage. **Required fields.** Actual and forecast series.

Interpretation. Scale-free, which makes it planning's shared currency for forecast accuracy. **Common failure.** Reporting it over a near-zero denominator. On the worked week a single storm-closed day of $400 turns a forecast whose typical daily miss is under three percent into a reported MAPE of 259.6 percent — one denominator, not seven bad forecasts, produced the number, and six of the seven days were still forecast within ten percent. **Verification.** Assert that every denominator is strictly positive and refuse to return a number otherwise. Print the minimum denominator and a count of thin days beside every MAPE displayed. The sentence that must accompany it: the denominator, not the forecast, produced the number.

Owned by. Chapter 10, Section 10.7.

B.11.4 Accuracy by horizon

Question. How does forecast error grow as the forecast reaches further out? **Formula.** MAE and MAPE computed separately within each block of the horizon, pooled across forecast origins.

Numerator. Errors within the horizon block. **Denominator.** Periods in the block times the number of origins. **Window.** Equal blocks of the forecast horizon, from each origin. **Filters.** The selection window's origins only. **Grain.** One horizon block, pooled across origins. **Unit.** The series' units, and percentage. **Required fields.** Actual and forecast series from multiple origins.

Interpretation. Expected error rises with horizon; realized error need not rise at every step. A flat or falling curve is a signal to inspect rather than a diagnosis: it can mean the origins sit too close together or the horizon is too short, and it can mean the series is genuinely no harder to forecast at range. **Common failure.** Quoting the count of daily errors as though they were independent observations, which overstates a weekly row's stability sevenfold. Report the number of independent origins separately from the number of daily errors. **Verification.** Report the spread across origins at each horizon beside the mean, and print whether the curve rises monotonically rather than assuming it.

Owned by. Chapter 10, Section 10.7.

B.11.5 Naive and seasonal-naive baselines

Question. What accuracy is available for free, before any model is fitted? **Formula.** The naive rule extends the last observed value flat across the horizon. The seasonal-naive rule repeats the last full seasonal period.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The same evaluation window as the candidate. **Filters.** A declared seasonal period — seven for daily data with a weekly rhythm. **Grain.** The series' own grain. **Unit.** The series' units. **Required fields.** The historical series and a declared period.

Interpretation. At a seven-day lag the seasonal-naive rule repeats last week, not last December. It carries the weekly rhythm for free and cannot climb a trend. **Common failure.** Reporting a fitted model's accuracy without its margin over the seasonal-naive rule. A forecaster who cannot state that margin does not know whether the model is contributing anything. **Verification.** Compute both on identical origins. A margin smaller than the noise band is not yet a finding.

Owned by. Chapter 10, Section 10.5.

B.11.6 Prediction interval and coverage

Question. What range of outcomes does the model consider plausible, and did reality land inside it? **Formula.** Nominal coverage is the interval's stated probability. Realized coverage is the share of evaluated periods that landed inside the band.

Numerator. Evaluated periods falling inside the band. **Denominator.** Number of evaluated periods. **Window.** The evaluated horizon, from a stated number of forecast origins. **Filters.** Evaluated periods only. **Grain.** One period; also reported by horizon block where the periods permit it. **Unit.** Percentage for coverage; the series' units for width. **Required fields.** Forecast bounds and actuals.

Interpretation. Coverage alone is badly incomplete, because an arbitrarily wide interval achieves perfect coverage while saying nothing. Publish the width beside the coverage so that no one can buy the second by inflating the first. **Common failure.** Adding up the edges of interval bounds to get an interval for a total. A quantile of a sum is not the sum of the quantiles: aggregate the simulated paths first and take quantiles second. **Verification.** Report nominal coverage, realized coverage, mean width, the number of evaluated periods, and the number of independent origins together. Coverage of 94 percent on sixty-one dependent daily observations is a reason to investigate, not a verdict.

Owned by. Chapter 10, Section 10.8.

B.11.7 Worked example

Seven days of chain revenue against a forecast. In an ordinary week the mean absolute error is $314.29 and the mean absolute percentage error is 2.75 percent, with a minimum denominator of $9,400. Now close a store on the Tuesday: actual revenue for that day falls to $400 while the forecast still reads $7,600.

The mean absolute error moves honestly, to $1,314.29. The mean absolute percentage error explodes to 259.59 percent — because that single day's absolute percentage error is 7,200 divided by 400, or 1,800 percent, one term of a seven-term average. 6 of the 7 days were still forecast within ten percent. One denominator, not seven bad forecasts, produced the number, and that is the sentence that must accompany every mean absolute percentage error this appendix's entry permits.

B.12 Experiment and Causal-Evidence Measures

Experimental measures are the only ones in this appendix that license a causal verb, and they license it narrowly: among the tested population, in the tested window, at the tested dose. The family includes the observational comparison these measures replace, so that the two can be told apart on sight. The reporting conventions follow Kohavi et al. (2020).

B.12.1 Treatment and control conversion rate

Question. What share of each randomized arm converted? **Formula.** Assigned customers with at least one qualifying order in the outcome window / assigned customers in the arm.

Numerator. Assigned customers with at least one qualifying order in the window. **Denominator.** All assigned customers in the arm, whether or not the message reached them. **Window.** A stated number of days following each customer's assignment. **Filters.** Intention to treat: every randomized customer with an observed outcome stays in the arm they were assigned to. A delivery failure is operational evidence and is not by itself grounds for removing anyone. **Grain.** One customer, the unit of randomization. **Unit.** Proportion, reported as a percentage. **Required fields.** variant, converted, assign_date, customer_id.

Interpretation. A customer who converts three times in the window is one conversion in a rate whose denominator is customers. **Common failure.** Switching the numerator to orders, which changes the question without saying so. Separately, filtering on anything measured after assignment breaks the randomization that the whole design bought. **Verification.** Bounded in [0, 1]. Reconcile the assignment flow: randomized, outcome observed, and included in the primary analysis, with any shortfall named.

Owned by. Chapter 11, Section 11.6.

B.12.2 Absolute lift

Question. How much did the treatment change the outcome, on the metric's own scale? **Formula.** Treatment rate − control rate.

Numerator. The difference between the two arm rates. **Denominator.** None — it is a difference. **Window.** The stated outcome window. **Filters.** Intention to treat, both arms. **Grain.** One customer, reported for the experiment. **Unit.** Percentage points, when the metric is a rate. **Required fields.** Both arm rates and both arm sizes.

Interpretation. The quantity that scales to business volume: multiplying it by the number of customers reached gives the incremental count directly. StyleCraft's test recovers +1.80 points against a 6.00 percent control rate. **Common failure.** Reporting it without the control rate beside it, or reporting only the relative figure. Thirty percent of an unstated base is not a quantity. **Verification.** Report the absolute lift first in percentage points with the control rate stated beside it, then the relative lift, then the translation into incremental conversions and dollars.

Owned by. Chapter 11, Section 11.6.

B.12.3 Relative lift

Question. How large was the change against its own baseline? **Formula.** Absolute lift / control rate.

Numerator. The absolute lift. **Denominator.** The control rate. **Window.** The stated outcome window. **Filters.** Intention to treat, both arms. **Grain.** One experiment. **Unit.** Percentage change — not percentage points. **Required fields.** Both arm rates.

Interpretation. Systematically the larger and more persuasive of the two whenever the baseline rate is small. StyleCraft's 1.80-point lift is a 30 percent relative lift. **Common failure.** Treating transportability as a statistical fact. A relative lift transfers to a different baseline only if the effect really is multiplicative in the baseline, which is an empirical hope rather than a property of the statistic. **Verification.** Compute its interval as a ratio, never by rescaling the difference interval by the observed control rate; that treats an estimate as a known constant and produces an interval that is too narrow and wrongly centered.

Owned by. Chapter 11, Section 11.6.

B.12.4 Confidence interval on a difference in proportions

Question. What range of true effects is consistent with what we observed? **Formula.** estimate ± $z^* \times$ SE, with $z^* = 1.96$ for a conventional 95 percent interval, where the standard error of a difference in two proportions is built from each arm's own rate and size.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The stated outcome window. **Filters.** Intention to treat, both arms. **Grain.** One experiment. **Unit.** Percentage points. **Required fields.** Both arm conversion counts and both arm sizes.

Interpretation. StyleCraft's interval runs from 0.81 to 2.79 percentage points — a range of more than three to one in business terms, from a point estimate that looks precise. **Common failure.** Reporting the point estimate alone, or presenting an interval far narrower than the arm sizes support, which usually means the analysis slipped to a finer grain than the randomization. **Verification.** The unpooled standard error uses each arm's own rate and is the one behind the interval; the pooled version belongs to the test statistic. On StyleCraft's numbers they differ in the fourth decimal, which is why nobody notices when the wrong one is used.

Owned by. Chapter 11, Section 11.6.

B.12.5 Treatment dose

Question. How many times, and when, does a treated unit receive the intervention? **Formula.** A declared count and timing of exposures, stated as part of the intervention's definition.

Numerator. Not applicable — a design parameter. **Denominator.** Not applicable. **Window.** The assignment and outcome windows. **Filters.** Treated arm. **Grain.** One treated customer. **Unit.** Count of exposures. **Required fields.** A registration entry; it cannot be recovered from the outcome data.

Interpretation. The dose fixes the cost per assigned customer and therefore the break-even lift. An experiment run at one alert per drop licenses a rollout at one alert per drop. **Common failure.** Rolling out at a different frequency. That is a different intervention whose economics and whose response must both be re-derived; at two alerts StyleCraft's break-even doubles to 0.86 points and the same measured interval no longer clears it at its lower end. **Verification.** Declare it before the test runs, and quote it in the same sentence as any economic conclusion.

Owned by. Chapter 11, Section 11.7.

B.12.6 Break-even lift

Question. What lift would this program need just to pay for itself? **Formula.** Cost per assigned customer / contribution per incremental conversion, where cost per assigned customer is cost per exposure times the dose.

Numerator. Cost per assigned customer. **Denominator.** Contribution per incremental conversion. **Window.** The program's own window. **Filters.** Assigned customers, at the tested dose. **Grain.** One program at one dose. **Unit.** Percentage points. **Required fields.** A costed exposure, a declared dose, and a contribution estimate.

Interpretation. StyleCraft's Drop Alert costs $0.12 per assigned customer against $28 of contribution per incremental conversion, so the break-even lift is 0.43 percentage points. **Common failure.** Comparing it against the point estimate. The comparison is made against the whole interval: if the lower bound clears the line the program pays even in the pessimistic world the data still permits; if the line falls inside the interval, the honest report is that the evidence does not settle the question. **Verification.** Undefined until the dose is stated. Publish a sensitivity grid across plausible contribution figures and doses, flagging any cell that falls inside the confidence interval.

Owned by. Chapter 11, Section 11.7.

B.12.7 Incremental conversions and net contribution

Question. What is the program worth per thousand customers at the tested dose? **Formula.** Incremental conversions = absolute lift expressed as a proportion × 1,000; equivalently, absolute lift expressed in percentage points × 10. Net contribution = incremental conversions × contribution per conversion − 1,000 × cost per assigned customer.

Numerator. Contribution earned on incremental conversions, less program cost. **Denominator.** Not applicable; the per-thousand scaling is the reporting convention. **Window.** The program's own window. **Filters.** Enrolled customers at the tested dose. **Grain.** One thousand customers enrolled. **Unit.** Dollars per thousand customers. **Required fields.** The lift and its interval, a contribution estimate, and a costed dose.

Interpretation. On StyleCraft's numbers: at the point estimate, 18.0 incremental conversions worth $504.00 against $120.00 of cost, a net of $384.00; at the interval's pessimistic edge, $106.80; at its optimistic edge, $661.20. The decision does not hinge on which end proves true. **Common failure.** Attaching the phrase 'at any scale'. The arithmetic assumes cost is linear in customers reached, contribution per conversion is stable, no fixed implementation cost amortizes, no capacity or inventory constraint binds, unsubscribe and fatigue costs are zero, and response does not diminish as the program scales. Several of those are known to weaken at rollout. **Verification.**

Compute at all three points of the interval, not only the estimate. Scrutinize the contribution figure: it is an average across observed orders, and the orders an incremental converter places need not carry the same mix or margin.

Owned by. Chapter 11, Section 11.7.

B.12.8 Observational difference (targeted versus untargeted)

Question. How did the customers who received the program differ from those who did not? **Formula.** Rate among those who received − rate among those who did not.

Numerator. The difference between the two observed group rates. **Denominator.** None — it is a difference. **Window.** Stated period. **Filters.** Groups formed by whatever mechanism assigned the treatment in the record — which is precisely what is unknown. **Grain.** One customer, reported for the two groups. **Unit.** Percentage points. **Required fields.** The outcome and a treatment indicator.

Interpretation. A true fact about the records and a false answer to the causal question. StyleCraft's observational difference of 5.30 points decomposes exactly into a 1.80-point within-stratum difference and a 3.50-point composition component — two thirds of the headline was who got the treatment. **Common failure.** Reporting it with a causal verb. Before quoting any comparison as evidence about a decision, state how each side was selected and what would have made the two sides differ even if the treatment had never existed. **Verification.** Where a confounder happens to be recorded, standardize on the treated group's own mix and report the composition component beside the adjusted difference. The two components must sum exactly to the naive difference.

Common variants. An adjusted difference, standardized on measured covariates, is stronger evidence and is still not a causal effect: adjustment corrects only the imbalances the analyst can name.

Owned by. Chapter 11, Section 11.3.

B.12.9 Statistical power and minimum detectable effect

Question. Is this test large enough to find an effect worth acting on? **Formula.** Power is the probability of detecting an effect of a stated size at a stated significance level and sample size; the minimum detectable effect is the smallest effect the design can reliably detect.

Numerator. Not applicable. **Denominator.** Not applicable. **Window.** The planned test duration. **Filters.** Per arm, two-sided, at a declared significance level. **Grain.** One customer per arm. **Unit.** Probability for power; percentage points for the minimum detectable effect. **Required fields.** A baseline rate from history, a declared minimum detectable effect, and a target power.

Interpretation. The minimum detectable effect is a business judgment, not a statistic. StyleCraft registers 1.5 points, needs 4,379 per arm, and runs 5,000. **Common failure.** Running underpowered and reading a null as no effect. Required sample size grows roughly with the inverse square of the smallest effect worth detecting — at 0.5 points the same design would need 36,778 per arm — and an underpowered test's significant results systematically overstate the effect's size. **Verification.** Register the baseline, the minimum detectable effect, the significance level, and the power before the test runs.

Owned by. Chapter 11, Section 11.5.

B.12.10 Difference-in-differences

Question. What changed in the treated group beyond what changed anyway? **Formula.** The treated group's change from before to after, less the comparison group's change over the same periods.

Numerator. Treated change less comparison change. **Denominator.** None — it is a difference of differences. **Window.** Equal pre and post periods for both groups. **Filters.** A comparison group chosen before the outcome is seen. **Grain.** The unit at which the treatment was applied — market, store, or customer. **Unit.** The outcome's own units. **Required fields.** Outcomes for both groups in both periods.

Interpretation. Rests on parallel trends, which cannot be verified, only supported. A pre-post report with no comparison group would have credited the campaign with the whole treated change. **Common failure.** Reporting the pre-post difference alone. It is a causal claim with no comparison group, whose implicit counterfactual is that nothing else would have changed. **Verification.** Plot both series over a long pre-period and inspect whether they

moved together before the intervention. License only the sentence that begins ‘under the stated parallel-trends assumption’.

Owned by. Chapter 11, Section 11.9.

B.12.11 Worked example

StyleCraft’s Drop Alert test randomized 5,000 customers to each arm. The control arm converted at 6.00% and the treatment arm at 7.80%, an absolute lift of 1.80 percentage points and a relative lift of 30%. The standard error of the difference is 0.5066 percentage points, giving a 95 percent confidence interval on the difference of 0.81 to 2.79 percentage points.

The decision is made against the whole interval rather than the point estimate. At one alert per customer the program costs $0.12 per assigned customer against $28 of contribution per incremental conversion, so the break-even lift is 0.43 percentage points. The entire interval sits above that line, so the program pays even in the pessimistic world the data still permits. Per thousand customers enrolled, the point estimate is 18.0 incremental conversions worth $384.00 net; the interval’s edges give $106.80 and $661.20.

Change the dose and the conclusion changes on identical evidence. At two alerts the break-even lift doubles to 0.86 percentage points, and the interval’s lower bound of 0.81 no longer clears it. This is why the treatment dose is an entry in this family rather than a footnote to one.

B.13 Dashboard and Decision Measures

This section creates no new formulas. It catalogs the measures that exist because a decision elevated them, and it records the apparatus by which any entry above becomes a dashboard KPI.

B.13.1 Weekly contribution

Question. How much contribution does this store generate each week toward covering its fixed footprint? **Formula.** Sum of (line_revenue − line_cost) within store and store-age week.

Numerator. Contribution on the store’s lines in the week. **Denominator.** None — it is a total. **Window.** One store-age week, on a scaffold that carries every open observable week. **Filters.** The declared store population; a week that is open and observable but carried no orders is zero, not missing. **Grain.** One store-week. **Unit.** Dollars per week. **Required fields.** line_revenue, line_cost, store_id, order_date, opening_date.

Interpretation. The unit in which a new store’s ramp is read against the fixed cost it must eventually carry. **Common failure.** Dropping the zero-order weeks. That removes the weakest observations from any benchmark computed over them and biases the benchmark upward, in the direction that makes a new store’s ramp look worse than it is. **Verification.** Reconcile the sum across store-weeks to the certified revenue and cost totals for the same population.

Owned by. Chapter 13, Section 13.12.1.

B.13.2 Four-week moving average of weekly contribution

Question. What is this store’s underlying weekly contribution, with single-week noise damped? **Formula.** The mean of weekly contribution over the current store-age week and the previous three, computed within store, ordered by store age, and returning missing until four observable weeks exist.

Numerator. Contribution over the trailing four store-weeks. **Denominator.** Four store-weeks. **Window.** Store-age weeks, from each store’s own opening. **Filters.** Grouped by store; ordered by store age ascending. **Grain.** One scaffold store-week. **Unit.** Dollars per week. **Required fields.** Weekly contribution on a complete store-week scaffold.

Interpretation. Every one of the five properties — the grouping, the ordering, the measure, the window, and the behavior before the window fills — has to be stated, because a moving calculation with any of them left to a default computes a different number and the difference is invisible in the output. **Common failure.** Computing it on a series with missing weeks, which averages the wrong four weeks. **Verification.** Confirm it returns missing for the first three weeks of each store rather than averaging a partial window.

Owned by. Chapter 13, Section 13.12.2.

B.13.3 Weekly fixed footprint (the coverage threshold)

Question. How much weekly contribution does a store need to carry its own fixed cost? **Formula.** The declared annual fixed footprint / 52. At StyleCraft, $92,000 / 52 = $1,769.23.

Numerator. The annual fixed footprint, a finance planning assumption. **Denominator.** 52 weeks. **Window.** Annual, expressed weekly. **Filters.** The population the finance assumption covers — new suburban and resort stores, with no figure supplied for the flagship format. **Grain.** One store-week; drawn as a constant reference line. **Unit.** Dollars per week. **Required fields.** A finance assumption, attributed to finance and declared before results are inspected.

Interpretation. Must be attributed to finance rather than to the data. A constant is correct here and a computed reference would be wrong. **Common failure.** Choosing it after the crossing weeks are known. A threshold chosen after the fact is not a benchmark but a description. **Verification.** Label the line with what it is, not with its value: weekly contribution carrying the $92,000 annual footprint, finance assumption, suburban and resort.

Owned by. Chapter 13, Section 13.12.1.

B.13.4 Weekly revenue at coverage

Question. How much weekly revenue does the footprint threshold correspond to? **Formula.** Weekly fixed footprint / contribution margin, on the same population. At StyleCraft, $1,769.23 / 0.46 = $3,846.15.

Numerator. The weekly fixed footprint. **Denominator.** The contribution margin of the same population. **Window.** Weekly. **Filters.** Suburban and resort stores, for both the threshold and the margin. **Grain.** One store-week. **Unit.** Dollars of revenue per week. **Required fields.** The threshold and a population-matched contribution margin.

Interpretation. A within-population conversion, and the only place that population's margin may be used. **Common failure.** Applying this population's margin to a chain-level quantity, or the chain's margin to this one. **Verification.** If the contribution margin moves by more than a point, the revenue threshold changes and every figure quoting it must be rebuilt before any title uses it.

Owned by. Chapter 13, Section 13.1.

B.13.5 First observed weekly coverage

Question. When did this store first cover its fixed footprint in a single week? **Formula.** The earliest store-age week in which weekly contribution reaches the declared weekly threshold.

Numerator. Not applicable — a week index. **Denominator.** Not applicable. **Window.** Store-age weeks from the store's own opening. **Filters.** Stores with enough observed history for the question to be meaningful. **Grain.** One store. **Unit.** Store-age week number. **Required fields.** Weekly contribution and the declared threshold.

Interpretation. A coverage milestone, not a payback period. It returns the same answer with or without the scaffold, because a week with no orders cannot be the first week above a positive threshold. **Common failure.** Reading it off a plotted line rather than computing it. A first-coverage week of 0 or 1 usually means the threshold was applied to revenue rather than to contribution. **Verification.** Compute it; do not read it. Confirm it is no later than the four-week-average coverage week for the same store.

Owned by. Chapter 13, Section 13.12.1.

B.13.6 First four-week-average coverage

Question. When did this store first reach its footprint on a smoothed basis? **Formula.** The earliest store-age week in which the four-week moving average of weekly contribution reaches the declared weekly threshold.

Numerator. Not applicable — a week index. **Denominator.** Not applicable. **Window.** Store-age weeks; the average returns missing until four observable weeks exist. **Filters.** The mature store population. **Grain.** One store. **Unit.** Store-age week number. **Required fields.** The four-week moving average and the declared threshold.

Interpretation. The go/no-go dashboard's decision measure, preferred to the raw crossing because it is harder to produce by accident. **Common failure.** Describing a smoothed first crossing in the language of the stricter one. It

is not a claim that the store held coverage, stayed above the line, or established durable coverage. **Verification.** A four-week-average coverage week earlier than the same store's observed week is arithmetically impossible; assert it. The gap between the two is itself readable: a store whose weeks are close crossed on strength its neighboring weeks shared.

Owned by. Chapter 13, Section 13.12.1.

B.13.7 Predecessor range

Question. What range did comparable stores occupy at this same store age? **Formula.** The minimum and maximum of the four-week moving average across the mature comparable stores, computed separately at each store-age week.

Numerator. Not applicable — a pair of bounds. **Denominator.** Not applicable. **Window.** The store-age weeks the mature stores cover. **Filters.** The mature comparable stores only, on the smoothed measure. **Grain.** One store-age week, as a pair of bounds. **Unit.** Dollars per week. **Required fields.** The four-week moving average for each comparable store, by store age.

Interpretation. The observed minimum-to-maximum spread of a handful of stores, not a statistical prediction interval. **Common failure.** Comparing a smoothed band against a raw series. Identical axes do not make differently smoothed quantities comparable. A band that is flat across store age means the bounds were computed at table scope rather than per week. **Verification.** A minimum computed across stores at each week is biased upward wherever one store's zero weeks were dropped. Confirm the scaffold is complete before computing it.

Owned by. Chapter 13, Section 13.12.2.

B.13.8 Mature-store annual contribution

Question. What does a comparable store contribute in a full year? **Formula.** Trailing 52-week contribution for each mature comparable store over the year ending on the analysis date, then averaged across those stores.

Numerator. Trailing 52-week contribution per store. **Denominator.** The count of mature comparable stores. **Window.** The 52 weeks ending on the declared analysis date. **Filters.** The mature comparable stores only. **Grain.** One store-year, averaged. **Unit.** Dollars per year. **Required fields.** Weekly contribution over a full trailing year.

Interpretation. StyleCraft carries approximately $137,000, and uses it as the denominator that turns a contribution interval into store-equivalents. **Common failure.** Annualizing a weekly mean, which inherits whatever seasonality the sampled weeks happened to carry. **Verification.** Compute from a genuine trailing 52 weeks per store rather than from a weekly average multiplied by 52.

Owned by. Chapter 13, Section 13.9.

B.13.9 Interval expressed in store-equivalents

Question. How large is the remaining forecast uncertainty in a unit the reader already understands? **Formula.** The width of the open forecast range / mature-store annual contribution, end to end; half that around the estimate.

Numerator. The end-to-end width of the interval, in dollars of contribution. **Denominator.** Mature-store annual contribution, in dollars of contribution per year. **Window.** The still-open portion of the forecast horizon. **Filters.** Chain in the numerator; the mature comparable stores in the denominator — a mismatch that must be named on the canvas. **Grain.** One horizon total. **Unit.** Mature-store-equivalents. **Required fields.** The forecast interval bounds and the mature-store annual contribution.

Interpretation. StyleCraft's $240,000 open range is about 1.8 stores end to end, or roughly plus or minus 0.9 stores around the estimate. **Common failure.** Two wrong conversions are available and both look plausible: multiplying the interval by the contribution margin, and dividing it by the annual fixed footprint. Neither belongs on a canvas. There is no margin in the correct arithmetic, because both quantities are already contribution. **Verification.** Show the conversion in steps on the canvas rather than asserting the result, and name the denominator as well as using it. Where a conversion requires an assumption that cannot be established, change what is forecast rather than disclosing the assumption and proceeding.

Owned by. Chapter 13, Section 13.9.

B.13.10 Worked example

The go/no-go dashboard's threshold is a worked example of the whole section. Finance carries an annual fixed footprint of $92,000 for a new suburban or resort store. Divided by 52 that is $1,769.23 of weekly contribution, which is the constant the primary view draws. Divided again by the population's 46% contribution margin it is $3,846.15 of weekly revenue — a within-population conversion, and the only place that margin may be used.

The forecast interval is the harder example. The still-open portion of the fiscal-year outlook spans $240,000 of contribution end to end. Divided by a mature store's annual contribution of $137,000, that is 1.8 store-equivalents end to end, or roughly plus or minus 0.9 stores around the estimate. Note what is absent from that arithmetic: there is no margin in it, because both quantities are already contribution. Two wrong conversions are available and both look plausible — multiplying the range by the contribution margin, or dividing it by the fixed footprint — and neither belongs on a canvas.

Table B.11 records the apparatus by which a metric from any earlier section becomes a measure on that dashboard. It is worth noting what the apparatus does not include: the chapters name no per-metric owner. Ownership in Chapter 13 attaches to the decision, to the analyst who signs the artifact, and to the artifact's refresh process — which is a deliberate narrowing, not an omission.

Table B.11

How a metric becomes a measure on a decision artifact

Element	What it requires	Where the guide sets it
Tier	Primary, supporting, or diagnostic, by proximity to the decision — and expressed in size and position, because a hierarchy stated in the analyst's head is not communicated at all	Section 13.4
Reader's question	The question this view answers for this reader, written out	Table 13.2
Measure and grain	The measure, and what one mark represents	Table 13.8
Reference carried	The baseline, target, comparison group, or benchmark drawn on the view. If the comparison is against a target, draw the target.	Section 12.9
Decision threshold	An externally supplied cut, declared before the results are inspected, because a threshold chosen afterward is not a benchmark but a description	Sections 12.9 and 13.12.1
Review cadence	When the artifact refreshes and when the decision is revisited	Section 13.13
Reversal condition	Two or three observable events that would change the recommendation, each with a date or a threshold attached	Section 13.12.7

B.14 Metric Index and Exclusion Register

The catalog above is ordered by family. This section supplies the two indexes a reader needs when the family is not obvious: an alphabetical list of every cataloged metric with the section that defines it and the chapter that owns it, and a register of the metrics that were considered and deliberately left out. The second list is the more important of the two. A dictionary that silently omits a metric looks identical to one that never considered it, and a reader who cannot find a definition here needs to know which of those two situations they are in.

B.14.1 Alphabetical index of cataloged metrics

Table B.12 lists every entry in Sections B.2 through B.13. Where the owning chapter column reads Appendix B, the measure is used by the guide or required by a project but is defined nowhere else, and this appendix is its only source.

Table B.12

Every cataloged metric, its entry, and its owning chapter

Metric	Entry	Owned by	Metric	Entry	Owned by
Absolute lift	B.12.2	Ch. 11 §11.6	Mean absolute error (MAE)	B.9.6	Ch. 8 §8.6
Accuracy	B.10.3	Ch. 9 §9.6	Mean absolute percentage error (MAPE)	B.11.3	Ch. 10 §10.7
Accuracy by horizon	B.11.4	Ch. 10 §10.7	Mean squared error (MSE)	B.9.8	Appendix B
Adjusted Rand index (ARI)	B.8.7	Ch. 6 §6.10	Monetary value (RFM)	B.3.9	Ch. 6 §6.5
App-user share	B.6.8	Appendix B	Naive and seasonal-naive baselines	B.11.5	Ch. 10 §10.5
Attributed orders	B.5.10	Ch. 5 §5.8	Net contribution after advertising spend	B.2.8	Ch. 1, Lab 1.2
Attribution (conversion) window	B.5.11	Ch. 3 §3.12.3	Net revenue	B.2.4	Ch. 4 §4.8
Average of group average order values (do not use)	B.3.2	Ch. 5 §5.3	Number of customers	B.2.11	Ch. 4 §4.8
Average order value (AOV)	B.3.1	Ch. 5 §5.7	Number of orders	B.2.10	Ch. 4 §4.8
Average order value decomposition	B.3.5	Ch. 5 §5.7	Observational difference (targeted versus untargeted)	B.12.8	Ch. 11 §11.3
Baseline score	B.9.9	Ch. 8 §8.7	Occasionwear share	B.3.16	Ch. 5 §5.7
Base rate	B.10.1	Ch. 9 §9.2	Odds ratio	B.10.15	Ch. 9 §9.4
Bounce rate	B.6.2	Appendix B	Open rate	B.6.3	Ch. 5 §5.7
Break-even lift	B.12.6	Ch. 11 §11.7	Opt-in rate	B.6.7	Ch. 3 §3.4.1
Calibration (reliability) check	B.10.9	Ch. 9 §9.8	Orders per customer (purchase frequency)	B.3.7	Ch. 6 §6.5
Capture at a selection share	B.9.11	Ch. 8 §8.6	Orders per month	B.3.12	Ch. 6 §6.6
Churn label	B.4.4	Ch. 9 §9.3	Pearson correlation coefficient (r)	B.9.1	Ch. 7 §7.3
Churn rate (descriptive)	B.4.3	Ch. 5 §5.7	Percentage change and percentage-point change	B.2.14	Ch. 13 §13.15
Click rate	B.6.4	Ch. 5 §5.7	Post-click conversion rate	B.5.4	Ch. 1 §1.10
Clicks	B.5.2	Ch. 1 §1.10	Precision	B.10.4	Ch. 9 §9.7
Click-through rate (CTR)	B.5.3	Ch. 1 §1.10	Predecessor range	B.13.7	Ch. 13 §13.12.2
Click-to-open rate (CTOR)	B.6.5	Appendix B	Prediction interval and coverage	B.11.6	Ch. 10 §10.8
Cluster centroid	B.8.9	Ch. 6 §6.10	p-value	B.9.5	Ch. 7 §7.10
Cohort retention rate (period-active)	B.4.5	Ch. 5 §5.8	Recall (true positive rate)	B.10.5	Ch. 9 §9.7
Cohort size	B.4.6	Ch. 5 §5.8	Recency	B.3.10	Ch. 6 §6.5
Confidence interval for a coefficient	B.9.4	Ch. 7 §7.10	Regression coefficient	B.9.3	Ch. 7 §7.8
Confidence interval on a difference in proportions	B.12.4	Ch. 11 §11.6	Relative lift	B.12.3	Ch. 11 §11.6

Metric	Entry	Owned by	Metric	Entry	Owned by
Confusion-matrix counts (TP, FP, FN, TN)	B.10.2	Ch. 9 §9.6	Repeat-order share	B.4.2	Ch. 12 §12.13.2
Contribution	B.2.6	Ch. 13 §13.12.1	Repeat purchase rate	B.4.1	Ch. 3 §3.10.3
Contribution margin	B.2.7	Ch. 13 §13.12.1	Return on ad spend (ROAS)	B.5.9	Ch. 1 §1.10
Cost per click (CPC)	B.5.5	Ch. 5 §5.7	Revenue growth rate	B.2.12	Ch. 2 §2.7
Cost per conversion	B.5.7	Ch. 1, Exercise 1.4	Revenue per customer	B.3.6	Ch. 5 §5.7
Cost per thousand impressions (CPM)	B.5.6	Appendix B	Revenue per exposure-month	B.3.13	Ch. 6 §6.6
Cumulative gains	B.10.14	Ch. 9 §9.10	Revenue per unit (realized price per unit)	B.3.4	Ch. 5 §5.7
Cumulative (overall) conversion rate	B.7.3	Ch. 5 §5.8	Revenue share by decile	B.3.17	Ch. 5 §5.9
Customer acquisition cost (CAC)	B.5.8	Ch. 5 §5.7	RFM grid region	B.8.5	Ch. 6 §6.5
Decision threshold	B.10.10	Ch. 9 §9.9	RFM scores (R, F, M)	B.8.4	Ch. 6 §6.5
Delivery rate	B.6.1	Appendix B	ROC AUC	B.10.8	Ch. 9 §9.8
Difference-in-differences	B.12.10	Ch. 11 §11.9	Root mean squared error (RMSE)	B.9.7	Ch. 8 §8.6
Discount dollars	B.2.3	Appendix B	R-squared	B.9.2	Ch. 7 §7.7
Discount share	B.3.14	Ch. 6 §6.6	Segment share of customers	B.8.2	Ch. 6 §6.11
Drop-off rate	B.7.4	Appendix B	Segment share of revenue	B.8.3	Ch. 6 §6.11
Expected net value	B.10.12	Ch. 9 §9.9	Segment size	B.8.1	Ch. 6 §6.10
Exposure months	B.3.11	Ch. 6 §6.6	Selection share	B.9.12	Ch. 8 §8.6
F1 score	B.10.6	Ch. 9 §9.7	Silhouette score	B.8.6	Ch. 6 §6.9
False positive rate and false negative rate	B.10.7	Ch. 9 §9.12.10	Stage count	B.7.1	Ch. 5 §5.8
First four-week-average coverage	B.13.6	Ch. 13 §13.12.1	Stage-to-stage conversion rate	B.7.2	Ch. 5 §5.8
First observed weekly coverage	B.13.5	Ch. 13 §13.12.1	Statistical power and minimum detectable effect	B.12.9	Ch. 11 §11.5
Forecast bias (mean signed error)	B.11.2	Ch. 10	Store share	B.3.15	Ch. 6 §6.6
Forecast error (signed)	B.11.1	Ch. 10 §10.7	Tenure	B.4.7	Ch. 6 §6.6
Four-week moving average of weekly contribution	B.13.2	Ch. 13 §13.12.2	Time to second purchase	B.4.8	Ch. 5 §5.7
Frequency (RFM)	B.3.8	Ch. 6 §6.5	Treatment and control conversion rate	B.12.1	Ch. 11 §11.6
Gross sales	B.2.2	Appendix B	Treatment dose	B.12.5	Ch. 11 §11.7
Impressions	B.5.1	Ch. 1 §1.10	Treatment rate (treated share)	B.10.11	Ch. 9 §9.9
Incremental conversions and net contribution	B.12.7	Ch. 11 §11.7	Units per order (basket size)	B.3.3	Ch. 5 §5.7
Inertia (within-cluster sum of squares)	B.8.8	Ch. 6 §6.8	Units sold	B.2.9	Ch. 5 §5.7

Metric	Entry	Owned by	Metric	Entry	Owned by
Interval expressed in store-equivalents	B.13.9	Ch. 13 §13.9	Unsubscribe rate	B.6.6	Ch. 11, Table 11.3, where it is named as an instrumentation obligation
Lift by decile	B.10.13	Ch. 9 §9.10	Variance from target or baseline	B.2.13	Ch. 2 §2.7
Line cost	B.2.5	Ch. 13 §13.12.1	Weekly contribution	B.13.1	Ch. 13 §13.12.1
Line revenue	B.2.1	Ch. 4 §4.8	Weekly fixed footprint (the coverage threshold)	B.13.3	Ch. 13 §13.12.1
Margin over baseline	B.9.10	Ch. 8 §8.9	Weekly revenue at coverage	B.13.4	Ch. 13 §13.1
Mature-store annual contribution	B.13.8	Ch. 13 §13.9			

B.14.2 Metrics considered and not cataloged

The inclusion rule for this appendix is strict: a metric belongs here when it is used in the guide, in the labs and exercises, or in one of the two projects. A useful but unused metric is not added to make the catalog appear comprehensive, because every entry is a commitment to maintain a definition. Table B.13 records what was considered and excluded, and why. Several of these entries are ones a reader will expect to find, which is precisely why the register exists.

Table B.13

The exclusion register

Measure	Why it is not cataloged
Reach and media frequency	Neither appears in Chapters 1-13 or in either project. StyleCraft's campaigns table records impressions and clicks but no de-duplicated audience, so reach cannot be computed and frequency cannot be derived. Note the name collision: the book's frequency is the RFM count of orders per customer, not exposures per person.
Segment index versus the total population	Absent from Chapter 6 and from every other chapter. The role an index would play is occupied by segment share of customers, segment share of revenue, and standardized centroids read in standard-deviation units. Adding an index here would introduce a measure the chapters do not teach.
Lift versus base rate as a segmentation measure	The book's only lift is Chapter 9's lift by decile, which is cataloged in Section B.10. Chapter 6 contains no lift measure of any kind; the word appears there only inside the list of causal verbs the chapter forbids.
Customer lifetime value (CLV)	Named in Chapter 1's exercises and listed among the columns customer_features ships, but no chapter defines a formula, a horizon, or a discount rate for it. Cataloging one here would invent a definition the book does not teach. Where a project requires it, the student defines it and records the definition using the form in Section B.1.
App adoption or activation rate over a window	app_user is a stored customer attribute overwritten in place, not an event series, so a rate of adoption across a period cannot be computed from the certified data. The point-in-time stock is cataloged as app-user share in Section B.6.
Engagement rate as a single measure	Excluded deliberately. The book uses channel-specific rates — open, click, click-to-open — and a combined engagement rate would obscure the denominators those entries exist to make explicit.
Net promoter score, satisfaction, and brand-awareness measures	No survey instrument ships with StyleCraft and no chapter computes one. Chapter 3 discusses construct validity for such measures without cataloging them.
Brier score, expected calibration error, and log loss	No numeric calibration summary statistic appears anywhere in Chapter 9. The chapter's calibration evidence is the per-bin difference and its worst value near the operating threshold, which is cataloged in Section B.10.
Weighted MAPE, symmetric MAPE, and scaled error measures	Chapter 10 evaluates on MAE, MAPE, and signed error only. The alternatives exist to repair MAPE's near-zero failure, and the chapter's chosen repair is to disqualify MAPE for series that approach zero rather than to substitute a different statistic.

Measure	Why it is not cataloged
Cost per thousand impressions in the chapters	Included in Section B.5 as an Appendix B entry rather than a chapter-owned one. The book computes spend and impressions on the same campaigns but never names the ratio; it is cataloged because both projects may quote it and a student computing it needs the denominator convention.
Return rate, cancellation rate, and net-of-returns revenue	StyleCraft ships no returns table. Several entries in Section B.2 note where returns would change the number; none can be computed from the certified data as it stands.

Two of these exclusions carry a standing obligation. Customer lifetime value and the survey-based measures are both quantities a project may legitimately need; when a project defines one, the definition belongs in that project's own dictionary on the form in Table B.3, not in a local variation of an entry here. And because the guide's completion rule requires that this appendix be updated whenever a chapter uses a metric it does not already carry, this register is the place a future revision records its reasoning rather than leaving a silent gap.

References

The works below are the sources for the definitional conventions this appendix adopts, rather than for the individual formulas, which are the guide's own and are cited to their owning chapters in each entry.

Campbell, D. T. (1979). Assessing the impact of planned social change. *Evaluation and Program Planning, 2*(1), 67–90. https://doi.org/10.1016/0149-7189(79)90048-X

Farris, P. W., Bendle, N. T., Pfeifer, P. E., & Reibstein, D. J. (2010). *Marketing metrics: The definitive guide to measuring marketing performance* (2nd ed.). Pearson Education.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

Kohavi, R., Tang, D., & Xu, Y. (2020). *Trustworthy online controlled experiments: A practical guide to A/B testing*. Cambridge University Press.

Parmenter, D. (2015). *Key performance indicators: Developing, implementing, and using winning KPIs* (3rd ed.). Wiley.

Schmittlein, D. C., Cooper, L. G., & Morrison, D. G. (1993). Truth in concentration in the land of (80/20) laws. *Marketing Science, 12*(2), 167–183. https://doi.org/10.1287/mksc.12.2.167

U.S. Securities and Exchange Commission. (2020). *Commission guidance on management's discussion and analysis of financial condition and results of operations* (Release Nos. 33-10751; 34-88094). https://www.sec.gov/rules/interp/2020/33-10751.pdf

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of Management Information Systems, 12*(4), 5–33. https://doi.org/10.1080/07421222.1996.11518099

Appendix C

AI Prompting Templates for Analytics

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · August 2026
Except where otherwise noted, this appendix is licensed under CC BY 4.0.

Appendix Information

PURPOSE

This appendix collects the prompting templates the guide uses for recurring analytics tasks. Its purpose is not to help you obtain polished output quickly; it is to help you specify a task precisely enough that the output can be predicted, tested, explained, and documented. Each entry is a copy-and-adapt structure that makes the analyst state the specification, the grain, the constraints, and the expected output before an assistant produces anything, and that requires the assistant to expose missing information rather than fill it.

VERSION AND DATE

Version 1.0 · August 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). AI prompting templates for analytics. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Appendix C, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Except where otherwise noted, this appendix is licensed under a Creative Commons Attribution 4.0 International License.
Copyright © 2026 by Jose Mendoza.

ChatGPT is a product of OpenAI. Claude is a product of Anthropic. Gemini, Google Colab, Google Drive, and NotebookLM are products of Google LLC. GitHub Copilot is a product of GitHub, Inc. Tableau, Tableau Desktop, and Tableau Prep are trademarks of Salesforce, Inc. “Python” and the Python logos are trademarks or registered trademarks of the Python Software Foundation. pandas, Matplotlib, NumPy, and statsmodels are NumFOCUS fiscally sponsored projects; scikit-learn is a NumFOCUS-affiliated project. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this appendix, including outlining, preliminary drafts, revision of prose, and document formatting. These tools were used under the author’s direction and are not credited as authors, researchers, or sources. The author determined the appendix’s scope, boundaries, and content, and reviewed and approved all AI-assisted material: factual claims and citations were checked against the underlying sources rather than accepted from AI-generated summaries, and every numerical output used in the companion worked examples was generated from the underlying code and checked against a stated expectation rather than entered manually. Responsibility for the accuracy, originality, and final form of this appendix rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

COMPANION FILES

An editable Markdown and Word template library, a plain-text version optimized for copying, a chapter-indexed prompt finder, and one completed worked example per major workflow are in the Applied Business Analytics companion repository.

How to Read a Template

The appendix contains twelve numbered sections. Sections C.1 through C.4 are the apparatus every template assumes — the division of labor, the prompt anatomy, the audit stack, and the privacy rule. Sections C.5 through C.10 hold the library. Sections C.11 and C.12 hold the anti-patterns and the index.

- C.1 How This Library Works

- C.2 The Common Prompt Anatomy
- C.3 The Audit Stack
- C.4 Privacy and Data Handling
- C.5 Specification and Measurement Templates (Templates C.1–C.4)
- C.6 Preparation and Description Templates (Templates C.5–C.8)
- C.7 Segmentation, Drivers, and Prediction Templates (Templates C.9–C.12)
- C.8 Forecasting and Experiment Templates (Templates C.13–C.15)
- C.9 Visualization and Communication Templates (Templates C.16–C.18)
- C.10 Review and Close (Template C.19)
- C.11 Anti-Patterns
- C.12 Template Index and Exclusion Register
- References

Every entry opens with an **Owned by** line naming the chapter and section it is drawn from, and then runs through the same six working parts, in the same order. The parts are labeled in bold rather than given headings, so that an entry stays on one or two pages and can be scanned while you work. Table C.1 states what the metadata line and each of the six parts give you.

Two labels carry numbers. **Template C.1** through **Template C.19** identify the entries themselves and are the labels a chapter cites. **Prompt C.1** onward identify the individual copy-and-adapt blocks, which run continuously through the appendix because several templates need more than one prompt. Sections are numbered separately, C.1 through C.12, so Section C.7 is where Template C.9 lives. Tables follow the same pattern: Table C.1, Table C.2, and so on.

Text inside square brackets is a placeholder you replace. Everything outside the brackets is meant to be sent as written, including the sentences that prohibit something — those sentences are load-bearing, and Section C.2 explains why. The bracketed convention is the chapters' own: it first appears in Section 4.10's `[one cleaning step]` and Section 12.12's `[paste the schema]`.

Prompts are set in a monospaced block so that line breaks survive copying. Nothing in a prompt block is code; it is text to paste into an assistant.

Table C.1

The metadata line and the six working parts of a template entry

Part	What it gives you
Owned by (metadata)	The chapter and section the template is drawn from, and how closely its prompt follows the chapter's wording. It tells you which chapter to return to when a judgment call arises.
Use this template when	The task and the chapter that owns it. If your task is not the one described, the audit at the end will not fit it.
What you must decide first	The decisions that must exist before the prompt is sent. An assistant asked to make these decisions will make them, silently, and the template will have failed.
Copy-and-adapt template	The prompt, with bracketed placeholders. Send it as written apart from the placeholders.
Likely failure modes	What the assistant is most likely to get wrong on this task, drawn from the owning chapter's named failure modes. Read this before reading the response.
Audit after the response	The check that must run, and the table that governs it. Section C.3 maps every audit instrument in the guide.
What to record in Appendix D	What this template adds to the standard Appendix D record of Section C.3. Documentation is part of the task, not a step after it.

C.1 How This Library Works

The guide gives an AI assistant a specific and limited job. It may draft, propose, explain, debug, enumerate, or critique. It may not decide what the analysis is, what the data mean, which definition governs, which threshold applies, or what should be recommended. Every chapter from Chapter 2 onward states some version of that division, and every template in this appendix is built to enforce it at the point where it is most often lost, which is the moment the prompt is written.

The division matters because assistant failures in analytics are rarely obvious. A leaky pipeline does not usually look different from an honest one; it looks better. A summary computed on the wrong denominator arrives as a tidy table. A cluster solution fitted once and never refit reads as structure. Fluency is not reliable evidence of correctness, and the general term for confident, plausible, wrong output is *confabulation* (National Institute of Standards and Technology, 2024). Capability is also uneven in ways that are invisible from the outside, so strong performance on one task does not guarantee strong performance on an adjacent one (Dell'Acqua et al., 2023). Table C.2 records the division of labor the guide applies.

Table C.2

What may be delegated and what may not

The assistant may	The analyst must	Stated in
Draft code from written derivation rules and window boundaries	Write the derivation rules and the window boundaries	Sections 4.10, 6.12, 8.11
Enumerate candidates — questions, confounders, features, chart forms, titles	Choose among them, and treat every proposal as a candidate for the specification rather than as analysis	Sections 5.10, 6.12, 7.11, 12.12, 13.11
Compute what is fully specified — power, break-even, sensitivity tables, reconciliations	Declare the inputs before the computation, and never after seeing a result	Sections 9.9, 11.7, 11.11
Explain an error, a method, or an inherited cell	Understand the explanation well enough to defend it in the meeting that funds the work	Sections 1.9, 10.11
Argue against a finished result	Decide which objections survive, and record why the rest were discarded	Sections 2.8, 11.11, 12.12, 13.11
Draft prose from verified numbers	Own every verb and every number in the sentence, and sign it	Sections 7.11, 12.12, 13.11
Draft a candidate specification, definition, registration, or view specification from written inputs	Choose among the candidates, supply every rule the draft could not know, and own the version that governs the work	Sections 2.8, 3.9, 11.11, 12.12
Nothing that has not been framed as a draft or a candidate for your approval	Define the metric, choose the threshold, define the label, name the segment, set the axis range, and take responsibility for the recommendation	Sections 3.9, 6.10, 9.9, 12.12

The last two rows are read together and are easy to misread apart. Several templates in this appendix do ask an assistant to draft a specification, a metric definition, an experiment registration, or a set of view specifications, and each of the owning chapters authorizes it. What the final row forbids is the same act performed without a candidate frame around it. A drafted metric definition is a proposal the analyst selects from and signs; a metric definition the assistant chose silently is the failure Section 3.9 names. The difference is not the output. It is whether the prompt made the choice visible and whether the analyst made it.

C.1.1 The Two-Prompt Pattern

The pattern emerges in Chapter 6 and becomes standard from Chapter 7 onward, and the templates in Sections C.7 and C.8 all follow it. The first prompt asks for mechanics only and ends by forbidding narration. The analyst then runs the audit. Only when the work survives does a second prompt ask for prose, and that prompt names the verbs

the evidence licenses and forbids the ones it does not. Earlier templates use a single prompt, because the tasks they serve do not separate production from interpretation in the same way.

The order is the control. A response that arrives with its own interpretation attached is much harder to audit, because the interpretation tells you what to see before you have looked. Chapter 12 makes the same point about charts: a rendered chart is persuasive in a way its specification is not.

> **CONCEPT**
>
> **Mechanics First, Audit Between, Narration Last**
>
> The two-prompt pattern separates two things an assistant will otherwise do in one breath: producing an artifact and telling you what it means. Prompt one ends with a phrase equivalent to *narrate nothing yet*. The audit runs on the artifact alone. Prompt two receives only the audited numbers and is told which verbs the evidence licenses.
>
> Chapters 12 and 13 extend the pattern to five steps, because a chart and a sentence are produced by the same act that makes their claim. The five steps are: request, written prediction, build, audit and grade, then repair — or, in Chapter 13, the reader. Templates C.16 through C.18 carry that longer form.
>
> Source: Course concept developed for this guide, consolidating Sections 7.11, 8.11, 9.11, 10.11, 11.11, 12.12, and 13.11.

C.1.2 Predict Before You Send

Every template assumes the prediction discipline of Section 1.7. Before the response arrives, write down what you expect: the row count, the total, the sign, the plausible band, the number of candidates that will overclaim. The prediction is what converts reading a response into auditing one, and the gap between predicted and actual is the finding.

Three chapters make the prediction an explicit step rather than an assumption. Section 10.11 requires the rolling-origin error to be predicted before the pipeline runs, because an analyst who has computed the baselines already knows the plausible band and a result outside it is a finding about the pipeline rather than about the future. Section 11.11 requires the interval width to be predicted from the arm sizes, because an interval far narrower than predicted means the analysis grain slipped. Section 13.11 requires a prediction of how many drafted titles will overclaim, and in which of three shapes.

C.1.3 The Templates Are Modular

No template is a script. Each is a structure to adapt, and adapting it well means deleting the parts that do not apply to your task and adding the constraints your task needs.

Three controls recur throughout the library, although their wording changes by task: **use only supplied information**, **expose rather than fill missing information**, and **stop at the boundary of the delegated work**. The second appears as the literal word `UNSPECIFIED` where a template asks the assistant to complete a form, and as task-specific language elsewhere — *say which field is missing and stop*, *name any feature you cannot trace*, *say what else you need rather than guessing*. Whatever the wording, do not delete the control; if a template's phrasing does not fit your task, replace it with phrasing that does.

C.2 The Common Prompt Anatomy

Every template in this appendix is assembled from the same twelve components. Not all twelve appear in every prompt — a debugging request does not need a decision, and a title request does not need a split protocol — but a prompt that omits a component it needs is a prompt that has delegated that component silently. Table C.3 lists them.

Table C.3

The twelve components of an analytics prompt

Component	What you supply
Decision	The decision or deliverable the analysis supports. Without it the assistant optimizes for an interesting answer rather than a usable one.
Task	One bounded action. Two actions in one prompt produce an artifact whose faults cannot be attributed.
Data context	Table names, schema, derivation rules, definitions, certified totals, and a safe sample. Everything the assistant is allowed to know.
Unit and grain	What one input row represents and what one output row must represent. The single most productive line in most prompts.
Method	The method already chosen, or the alternatives to be compared. Never left open unless comparison is the task.
Constraints	Columns allowed, exclusions, reference categories, seeds, privacy limits, and course rules.
Expected output	The required format: a table, a code cell, a numbered list, twelve sentences. Stated as a shape, not as a wish.
Prediction	What the assistant must state before producing the artifact — row-count effects, totals, output grain, expected cardinality, evaluation figures.
Verification	The checks the output must print or explain, and the totals it must reconcile to.
Uncertainty	The instruction to write UNSPECIFIED for anything the supplied information does not determine, and to ask rather than fill.
Terminal prohibition	The clause that stops the assistant at the boundary of the task: narrate nothing yet; do not interpret; do not render conclusions; choose no threshold yourself.
Documentation	The reminder that the exchange is recorded under Appendix D, including the output you rejected.

Two of the twelve deserve their own note, because they are the ones students omit and the ones that do most of the work.

The **uncertainty** clause is the defense against invention. An assistant asked to fill a template will fill it; asked to define a metric with no context, it will produce a definition and will not say that it chose one; asked to design an experiment from an incomplete brief, it will supply a duration, a unit of randomization, and a treatment dose, silently and plausibly, so that the result reads like a plan you made. The clause that defeats this is short, and every template carries a version of it: *use only the fields and facts supplied below; where the information is insufficient, write UNSPECIFIED and ask me, and do not fill it yourself.* Every template that asks the assistant to complete a structure from supplied information carries a version of it; the templates that ask for a critique or a diagnosis carry the task-specific equivalent instead.

The **terminal prohibition** is the defense against scope drift. It is the sentence that keeps prompt one from becoming the whole analysis. The chapters supply the wording, and the wording is task-specific: *change nothing in the specification; narrate nothing yet* in Sections 7.11 and 8.11; *do not touch the test set, output scores not verdicts, choose no threshold yourself* in Section 9.11; *do not interpret* in Section 11.11; *do not render or describe conclusions* in Section 12.12.

CONCEPT

UNSPECIFIED Is the Load-Bearing Clause

The count of UNSPECIFIED rows a response returns is a measurement of your own prompt, not of the assistant. A registration draft that comes back with eight unspecified rows has told you that your brief was missing eight decisions, all of which someone would otherwise have made for you.

> A response with no unspecified rows against an incomplete brief has not answered your question. It has answered a question it invented, and the invention is invisible because it is written in the same voice as everything else.
>
> Source: Course concept developed for this guide, informed by Section 11.11.

C.3 The Audit Stack

A prompt is only half of a delegation. The other half is the audit that runs on the response, and the guide builds its audits cumulatively rather than replacing one with the next. Chapter 8 establishes a five-point audit — frame, leakage, split hygiene, baseline, and error in decision units — and every later chapter adds a supplement that runs with it rather than instead of it. Two chapters happen to label their supplements S1 through S4 — Table 9.5 for classification and Table 10.5 for forecasting — so cite the table as well as the point. Table C.4 maps every task in this appendix to the instrument that governs it.

Two entries in the table need a caution. In Chapter 12 only three of the five points transfer: *frame* becomes the question the view was built to answer, *baseline* becomes the reference the reader is meant to read values against, and *error in decision units* becomes whether the quantities are in units a decision maker can act on, while *leakage* and *split hygiene* have no visual analogue and are marked not applicable rather than skipped. In Chapter 11, *baseline* reads as the control arm and the break-even line.

Table C.4

Which audit governs which task

Task	Audit instrument	Defined in
Any analytic specification	The ten components, each present and none reconstructed after the results	Table 2.5
A data dictionary or metric definition	Allowed values checked against business rules rather than observed values; measurement levels corrected; the four elements plus grain and unit	Sections 3.5, 3.9; Appendix B
A cleaning step	The verification log: predicted and actual row count, distinct keys, totals, and missing counts, with every mismatch explained	Section 4.2.1
A descriptive table	Three moves — recompute one cell by hand, check printed totals against the log, strike every causal verb	Section 5.10
A segmentation	Four moves — verify one centroid by hand, read the stability indices, check the holdout profile against written predictions, strike unredeemable nouns from every name	Section 6.12
A regression	The four-point audit — signs, magnitudes, units, diagnostics	Table 7.5
A predictive pipeline	The five-point audit — frame, leakage, split hygiene, baseline, error in decision units	Table 8.5
A classifier	Table 8.5 plus Table 9.5's S1–S4 — base rate and floor, matrix arithmetic, threshold provenance, calibration and selection hygiene	Tables 8.5 and 9.5
A forecast	Table 8.5 plus Table 10.5's S1–S4 — time-aware evaluation, selection separated from the grade, calendar and breaks declared, metric and interval honesty	Tables 8.5 and 10.5
An experiment	Table 8.5 plus Table 11.7's E1–E3 — registration precedes analysis, assignment integrity, inference honesty	Tables 8.5 and 11.7
A chart or Tableau view	The three transferable points of Table 8.5 plus Table 12.5's V1–V3 — grain and aggregation, scale and baseline honesty, encoding fits the purpose	Tables 8.5 and 12.5; Appendix E

Task	Audit instrument	Defined in
A dashboard, title, or memo	Tables 8.5 and 12.5 plus Table 13.6's C1–C3 — every sentence is earned, the reader's takeaway matches the intended one, what was removed is disclosed	Tables 8.5, 12.5, and 13.6; Appendix E

C.3.1 The Record Every Exchange Leaves

Documentation is the fourth discipline of Section 1.7, and it is part of the task rather than a step after it. Every exchange run from this appendix leaves the same standard record, stated here once so that nineteen entries need not repeat it.

The standard record. The prompt as sent; the response disposition, meaning accepted, modified, or rejected; the specific error or limitation you found; the prediction you wrote before the check and the verification you then performed; and the change the exchange produced in your final work. Appendix D is the form this record is submitted on, and Table 1.6 is its compact notebook version.

Each entry's closing line names only what that template adds to the standard record — the leaderboard, the feature trace, the counted UNSPECIFIED rows, the clause-by-clause diff. Where an entry adds nothing, the standard record is the whole obligation.

One general check runs on every response in this appendix and belongs to no chapter in particular. It is Section 1.9's four questions, asked in order: does the output answer the question I specified; does one row still represent what I said it represents; can I reproduce at least one number by hand; can I state in one sentence what this result does not show. If any answer is no, the work is not ready to leave your desk.

C.4 Privacy and Data Handling

Every template in this appendix assumes that what you paste into an assistant is safe to paste. Deciding that is a measurement-custody decision rather than a convenience, and it is the analyst's, not the tool's. Table C.5 states what may be supplied.

Table C.5

What may and may not be supplied to an external AI assistant

May be supplied	Must not be supplied
Schemas, column names, data types, and levels of measurement	Customer records, order records, or any row describing an identifiable person
Derivation rules, metric definitions, and window boundaries	Names, email addresses, phone numbers, addresses, or account identifiers
Certified aggregate totals — row counts, distinct keys, revenue sums	Proprietary, contractual, or otherwise restricted institutional data
Course-provided synthetic StyleCraft files and the small practice files	Credentials, API keys, tokens, connection strings, or file paths that contain any of these
Small synthetic samples you constructed yourself	Confidential employer data, in any volume, absent an approved enterprise environment
Error messages and tracebacks, with paths and identifiers removed	Anything you are not authorized to disclose outside the approved course or organizational environment

The synthetic StyleCraft files are safe by construction; that is what they are for. Where a template asks you to paste sample rows, paste them from the course files. In professional work, the equivalent move is a schema plus a synthetic sample, or an approved enterprise tool, and the choice is recorded in the data and privacy declaration of Appendix D. Section 1.13 owns this rule and Section 3.13 extends it to measurement custody; this appendix only enforces it.

One further habit is worth adopting early. Record, at the time of the exchange, exactly what you supplied — not a later reconstruction. The declaration is easy to write honestly on the day and impossible to write honestly a week afterward.

C.5 Specification and Measurement Templates

The four templates in this section serve the framing work of Chapters 2 and 3. They are the templates most often skipped and the ones whose omission is most expensive, because a specification error propagates through everything built on it and is not detectable in any later output.

Template C.1 — Turn a Business Request into an Analytic Specification

Owned by. Chapter 2, Section 2.5. The ten components are Table 2.5's.

Use this template when. A stakeholder has made a request in business language and you need a first draft of the specification that will govern the work. The assistant is a brainstorming partner here, widening your view of the problem; it is not the author of the specification.

What you must decide first.

- The request as it was actually made, quoted rather than paraphrased. A paraphrase has already done half the framing.
- Which facts about the situation are established and which are your assumptions. Supply only the first.
- Whether you are willing to accept the request's framing. An assistant will usually accept the framing it is given, and asked to outline an analysis showing that a decision is working, it will comply.

Prompt C.1. Draft an analytic specification from a business request

```
CONTEXT
Business request, verbatim: [PASTE THE REQUEST AS IT WAS MADE]
Who made it: [ROLE]  Deadline stated or implied: [DATE OR NONE]
Established facts: [TWO OR THREE SENTENCES OF FACT ONLY]
Tables available, with grain: [TABLE = ONE ROW EQUALS ONE ...]
Fields available: [PASTE THE SCHEMA]

TASK
Draft a first version of an analytic specification with exactly these ten
components, one short paragraph each, and nothing else:
  1  Decision                     what choice will this analysis inform?
  2  Decision-maker and deadline  who decides, and by when?
  3  Analytical questions         what must be measured or compared?
  4  Unit of analysis             what does one row represent?
  5  Data                         what data, from where, what period?
  6  Candidate methods            what fits the questions and the deadline?
  7  Deliverable                  what will the decision-maker receive?
  8  Verification plan            how will the output be checked?
  9  Limitations and risks        what can this analysis not establish?
 10  AI-use plan                  where will AI assist, and how checked?

CONSTRAINTS
Use only the fields and facts supplied above.
Where the request does not determine a component, write UNSPECIFIED and
state the question you would need answered; do not fill it yourself.
Identify ambiguity. Do not resolve it by assumption.
Do not propose findings. Do not state what you expect the answer to be.
End with a numbered list of open questions for the requester.
```

Likely failure modes.

- Generic fluency — questions that would fit any retailer rather than this one, and no organizational constraint that no dataset records.

- Framing amplification — the plan advocates for the answer the request implies rather than testing it. Template C.2 exists for this reason and should be run on the result.
- A verification plan written as a sentiment rather than as checks with numbers attached.
- An AI-use plan that says AI will be used without saying how its output will be checked.

Audit after the response.

- Every one of the ten components present, and none of them reconstructed after the fact.
- The unit of analysis stated as *one row equals one what*, not as a topic.
- The verification plan naming quantities that can fail: row counts against a known roster, rate bounds, totals reconciled to a certified source, one row hand-checked.
- Every UNSPECIFIED row treated as a question for the requester, not as a gap for you to close quietly.

What to record in Appendix D. Beyond the standard record of Section C.3: the request as pasted, the assistant's draft, which components you rewrote entirely, which UNSPECIFIED rows you resolved and with whose answer.

Template C.2 — Review an Existing Analytic Specification Adversarially

Owned by. Chapter 2, Section 2.8. The request sentence is that section's, verbatim; the two closing controls are this appendix's.

Use this template when. You have a specification you believe in and want it attacked before it becomes expensive. Run this on your own work, including work you drafted with Template C.1.

What you must decide first.

- That you want a critique and not a rewrite. A rewritten plan transfers ownership of the framing to the assistant, which is the thing this template exists to prevent.
- What you expect the analysis to show. The prompt asks for an alternative explanation of that expected result, which requires you to have stated it.

Prompt C.2. Adversarial review of a specification

```
Here is my analysis plan.

[PASTE THE COMPLETED SPECIFICATION]

Here is what I expect the analysis to show: [ONE SENTENCE]

Act as a skeptical reviewer. Identify the weakest analytical question, one
stakeholder I have ignored, one alternative explanation for the result I
expect, and one verification check I have not planned.

Answer as a numbered critique. Do not rewrite the plan.
For each point, state what would have to be true for the point to matter.
Do not raise an objection that would apply equally to any analysis plan.
```

Likely failure modes.

- Generic objections that would apply to any plan — add more data, consider seasonality — and that cost nothing to raise.
- A rewrite disguised as a critique, in which the assistant supplies an improved plan rather than naming defects in yours.
- Objections that are unfalsifiable, so that acting on them cannot be evaluated.

Audit after the response.

- Take the critique point by point. Revise on at most two points, and reject at least one with a one-sentence justification. The justification is the graded artifact in Exercise 2.7 and the useful one in practice.
- For each point you accepted, state what would have to be true for it to matter, and whether it is true here.

What to record in Appendix D. Beyond the standard record of Section C.3: the specification version you submitted, the critique in full, which two points you acted on, which point you rejected and why.

Template C.3 — Draft or Audit a Data Dictionary Entry

Owned by. Chapter 3, Section 3.9. The instruction sentence is that section's, verbatim; the framing and the closing controls are generalized from its surrounding prose.

Use this template when. You need the tedious first draft of documentation for a table you are about to work with. This is close to the ideal delegation: the assistant is fast at the draft and structurally unable to supply the part that matters.

What you must decide first.

- Which sample rows you will paste, and that they are safe to paste. Use the course synthetic files; see Section C.4.
- Who owns the business rules for these columns. The draft has to be audited against them, and if nobody owns them, that is the finding.
- The table's grain, in your own words, before you ask.

Prompt C.3. Draft a data dictionary from a sample

```
Below is a small sample of rows from [TABLE NAME], which is course-provided
synthetic data. The table's grain is [ONE ROW EQUALS ONE ...].

[PASTE 5 TO 10 ROWS]

Draft a data dictionary entry for each column: definition, type, level of
measurement, allowed values, and one plausible known issue. Mark any column
where you are guessing.

For each column also state the table and grain it belongs to, the source
system or derivation rule if one is inferable, and what a missing value
would mean.

Where the sample cannot tell you the answer, write UNSPECIFIED rather than
inferring an allowed range from the values you can see. Do not treat the
observed values as the permitted values.
```

Likely failure modes.

- Inference from a sample presented as knowledge of a system. An assistant seeing discount values between 0 and 0.4 will document the allowed range as 0 to 0.4 when the business rule permits 0.8.
- Inherited silences. Nothing in a file reveals that a null identifier comes in two kinds, so no assistant will document it unprompted.
- Stored data type reported as level of measurement, which converts an identifier into a quantity and is among the most common silent errors in AI-generated analysis code.
- Generic known-issue guesses that read as diligence and contain nothing.

Audit after the response.

- Verify allowed values against business rules rather than observed values.
- Correct the measurement levels; the stored type is not the level.
- Add the issues only humans know, and mark the entry incomplete until a source owner has seen it.
- Predict the type and level of every column before you read the draft, and grade the draft against your predictions rather than the reverse.

What to record in Appendix D. Beyond the standard record of Section C.3: the sample you pasted and its source, the draft, at least one allowed-values range the assistant inferred from the sample rather than from a rule, at least one known issue it could not have known.

Template C.4 — Draft Candidate Metric Definitions for Analyst Selection

Owned by. Chapter 3, Section 3.5, using the entry schema of Appendix B. This entry implements the blueprint's *Define a Metric Completely* with the selection step made explicit: the assistant drafts candidates, the analyst defines the metric.

Use this template when. A metric is being used in your organization or your project and you cannot find a written definition of it. This template produces *candidate* definitions in the guide's canonical fields, which you select among and then take to whoever owns the number. Section 3.9 owns the reason the plural matters: an assistant asked to define a metric will produce a definition — some definition — and a definition that arrives fluently is easily mistaken for the organization's.

What you must decide first.

- Which decision the metric supports. A metric with no decision behind it cannot be evaluated, only computed.
- Whether Appendix B already carries the metric. If it does, use that entry; do not generate a competing definition.
- The window boundary convention you intend to use, inclusive or exclusive on each end. The assistant will pick one silently otherwise.

Prompt C.4. Define a metric in the canonical fields

```
Metric to define: [NAME AS IT IS USED IN THE ORGANIZATION]
Decision it supports: [DECISION]
Tables and fields available, with grain: [PASTE SCHEMA]
Window boundary convention: [INCLUSIVE / EXCLUSIVE ON EACH END]

Return the definition in exactly these fields, one line each, and nothing
else:
  Metric name          Business question     Formula in plain language
  Numerator            Denominator           Window, with its boundary rule
  Filters              Grain                 Unit
  Required fields      Interpretation        Common variants
  Verification

RULES
Use only the fields listed above. If the metric cannot be computed from
them, say which field is missing and stop.
Where more than one numerator, denominator, filter, window boundary, or
eligibility rule is defensible, return the alternatives separately under
Common variants. Do not select among them, and do not present one as the
definition.
Mark any element not determined by the business rules I supplied as
UNSPECIFIED rather than inferring it.
State whether the result is a proportion, a percentage, a ratio, a count, or
a currency amount.
Give a verification an analyst can perform by hand on ten rows.
Do not compute the metric. Do not interpret a value.
```

Likely failure modes.

- Fluent completion of ambiguity. Asked to define a metric with no context, an assistant produces a definition — some definition — and the fluency of its arrival is easily mistaken for authority.
- A denominator that conceals a grain: a rate that looks defined but is computed over order lines when the claim is about orders.
- A single definition offered where practice legitimately varies, with the choice unmentioned. This is the failure the template exists to prevent, and the reason it asks for candidates rather than an answer.
- A verification that restates the formula rather than testing it.

Audit after the response.

- Select among the variants yourself, in writing, and record the reason. A template that returns one variant where two are defensible has already failed.
- Check each of the five elements — numerator, denominator, window, filters, grain — against the decision the metric serves.
- Ask whether the assistant silently changed eligibility or mixed grains between the numerator and the denominator.
- Compute the metric by hand on ten rows and confirm the stated verification catches a deliberately wrong denominator.
- Reconcile the result against Appendix B if an entry exists, and record any divergence as a definitional finding rather than resolving it privately.

What to record in Appendix D. Beyond the standard record of Section C.3: the metric name as it was used in the request, the returned definition, the element you had to correct, whether the assistant disclosed choosing among variants.

C.6 Preparation and Description Templates

The four templates in this section serve Chapters 4 and 5, where the assistant is a competent junior pair of hands and the analyst is accountable for what the code did to the data. The division is sharp: in Chapter 4 the assistant drafts cleaning code and the analyst audits the counts; in Chapter 5 the assistant drafts summaries and the analyst audits the claims.

Template C.5 — Draft One Data-Cleaning Step

Owned by. Chapter 4, Section 4.10. The instruction block is that section's, verbatim, with placeholders in place of its example; the constraints below it are generalized from the same section.

Use this template when. You have decided what a cleaning step should do and want the code written. One step at a time; a prompt that asks for a pipeline gets a pipeline whose faults cannot be attributed to any step.

What you must decide first.

- Which remedy the step applies, by name. Drop, impute, or flag is an analytic decision with a bias attached, and Chapter 4 owns it. An assistant asked to "handle" missing values will default to one and not say so.
- The baseline quantities from your verification log: row count, distinct orders and customers, revenue total, missing counts in the affected columns.
- Your own prediction of what the step will change, written before the response arrives.

Prompt C.5. Draft one cleaning step with its predicted effects

```
DATA DICTIONARY ENTRIES for the affected columns:
[PASTE]

CERTIFIED BASELINE from my verification log:
  rows [N]   distinct orders [N]   distinct customers [N]
  revenue [$N]   missing values in the affected columns [N]

SAMPLE ROWS (course-provided synthetic data):
[PASTE 5 TO 10 ROWS]

Write pandas code to [ONE CLEANING STEP]. Before the code, state which
quantities the step should change — columns, rows affected, row count,
revenue total — and what the code deliberately leaves alone. Where the
baseline above determines an exact expected value, state it. Where it does
not, write UNSPECIFIED and include code that computes the pre-step figure
before modifying anything. If the step is a join, state the expected
cardinality and enforce it in the code. List any rows it would drop and why.

CONSTRAINTS
```

```
Do not perform any other cleaning step.
Do not modify the raw frame in place; assign to a new name.
Do not choose a missing-value remedy. I have chosen [DROP | IMPUTE | FLAG]
and you are implementing it.
Print the row count, the revenue total, and the affected missing-value
counts before and after.
Use only the columns described in the dictionary entries above.
```

Likely failure modes.

- Silent row drops. Coercion followed by a drop, a filter written wider than intended, or a grouping key containing nulls will shrink the file, and nothing announces it.
- Wrong join keys, wrong key types, or a dimension table that is not unique on the key — producing a result that runs, returns rows, and is quietly wrong.
- Unstated imputation defaults, usually mean or mode imputation, importing a bias without naming the choice.
- Editing the raw frame in place, which destroys the ability to re-run the log from its baseline.

Audit after the response.

- Compare every predicted quantity against the actual and explain every mismatch before continuing.
- If the assistant neither states the expected row-count effect nor explains why it is not determinable from the information you supplied, the audit has failed and the code should not be run. An honest `UNSPECIFIED` with a pre-step count in the code is a pass; silence is not.
- For a join, confirm the cardinality was both stated and enforced, and read the matched and unmatched counts.
- Record the step in the verification log whether it passed or not. A step that behaved unexpectedly and was explained is a log entry, not a failure.

What to record in Appendix D. Beyond the standard record of Section C.3: the step in one sentence, the predicted and actual quantities, the mismatch you investigated, whether the assistant disclosed its own remedy choice.

Template C.6 — Diagnose a Python Error

Owned by. This appendix. No chapter prints a debugging prompt; Section 1.9 names debugging as a legitimate use and Appendix A, Section A.14, catalogs the recurring errors by symptom.

Use this template when. A cell has failed and you want the error explained rather than the cell rewritten. The distinction matters: an assistant handed a broken cell will often return a working cell that does something else.

What you must decide first.

- What you expected the cell to produce. Without it, any code that runs looks like a fix.
- The smallest cell that still reproduces the error. Pasting a whole notebook invites a rewrite of the whole notebook.
- Whether the traceback or any path in it contains an identifier you should remove before pasting.

Prompt C.6. Diagnose an error without rewriting the cell

```
I am working in Google Colab with pandas.

EXACT ERROR MESSAGE (full traceback, identifiers removed):
[PASTE]

THE SMALLEST CELL THAT REPRODUCES IT:
[PASTE THE CELL]

RELEVANT DATA TYPES AND TWO SAMPLE ROWS:
[PASTE df.dtypes AND df.head(2)]

What I expected: [ONE SENTENCE]
What actually happened: [ONE SENTENCE]
```

```
Return, in this order and nothing else:
  1  a plain-language explanation of what the error means
  2  the single most likely cause, given what I pasted
  3  the smallest correction that would fix it
  4  one check confirming the correction did not change any output I did
     not intend to change

Do not rewrite the cell to do something different from what it does now.
Do not add cleaning, filtering, or aggregation I did not ask for.
If the traceback is insufficient to identify the cause, say what else you
need rather than guessing.
```

Likely failure modes.

- A rewrite that changes behavior. The cell runs; it no longer does what you wanted; nothing says so.
- Helpful additions — a dropna, a fillna, a filter — inserted to make the error go away rather than to explain it.
- A confident diagnosis of a cause the traceback does not support, when the real cause is upstream.
- A correction that fixes this cell and silently breaks a later one, which the assistant cannot see.

Audit after the response.

- Run the corrected cell and then re-run the cells that depend on it, checking that no unrelated output changed.
- Read the traceback yourself from the final line upward and confirm the assistant's reading of it, per Appendix A, Section A.3.
- If the correction added a step you did not ask for, remove it and see whether the error returns. If it does not, the added step was not the fix.

What to record in Appendix D. Beyond the standard record of Section C.3: the exact error, the minimal cell, the correction you accepted, the check confirming nothing else changed.

Template C.7 — Produce a Descriptive Table

Owned by. Chapter 5, Section 5.10. The instruction block is that section's, verbatim, with placeholders in place of its example; the constraints below it are generalized from the same section.

Use this template when. You know exactly which table you want and want the groupby-and-aggregate scaffold written. Descriptive work is where assistance is most fluent and, for that reason, most quietly dangerous: the failures are analytic decisions hiding inside the computation, and they arrive as a tidy table that looks finished.

What you must decide first.

- The exact table, stated as a shape: what one output row represents and which columns it carries.
- The denominator of every rate in it. If you cannot state the denominator, the table is not specified yet.
- The certified totals the table must reconcile to, from the Chapter 4 verification log.

Prompt C.7. Produce a descriptive table that must reconcile

```
DATA DICTIONARY ENTRIES:
[PASTE]

CERTIFIED TOTALS from my verification log:
  rows [N]   distinct orders [N]   distinct customers [N]
  total revenue [$N]

Write pandas code to produce [ONE SPECIFIC TABLE]. Before the code, state:
the unit of analysis of each output row; the exact denominator of every rate
or proportion; which measure of center you will use and why, given that
order value is right-skewed; and the reconciliation I should expect — which
totals of your table must equal the certified totals above. After the code,
print those reconciliation figures.

CONSTRAINTS
Do not narrate the table.
```

```
Do not use a causal verb anywhere in the response.
Do not average group means to produce an overall figure; weight them, and
state what you weighted by.
Report every rate with its denominator in the column name or a printed
note.
Use only the fields in the dictionary entries above. If the table requires
a field I have not supplied, say so and stop.
```

Likely failure modes.

- The silent mean, reached for on data known to be skewed, reporting a center the median contradicts.
- The invented denominator, chosen without disclosure, which retires the four-element discipline into a code comment.
- The averaged average — group means averaged to produce an overall figure that reconciles to nothing.
- The unreconciled table, whose filtered totals no longer match the certified file and whose filter is mentioned nowhere.
- Narrative overreach: asked to explain a table, an assistant supplies causes, converting description into speculation at exactly the step stakeholders quote.

Audit after the response.

- Recompute one cell by hand from the certified table.
- Check the printed reconciliation figures against the verification log.
- Strike every causal verb from any narration before it travels; descriptive language includes *is associated with*, *accounts for*, and *coincides with*.
- Confirm that any overall figure was weighted rather than averaged, and that the weight is named.

What to record in Appendix D. Beyond the standard record of Section C.3: the table requested, the stated denominator and center with the assistant's reason, the reconciliation figures, and any denominator or center you had to correct.

Template C.8 — Propose an Exploratory Analysis Plan

Owned by. Chapters 4 through 7 and 12, which each name candidate enumeration as a legitimate delegation. Section 5.10 sets the governing condition: proposals are candidates for the specification, not analysis.

Use this template when. You are before the modeling stage and want a wider slate of views than you would generate unaided. Enumerating rivals is a breadth task, and breadth is what an assistant has.

What you must decide first.

- What has already been established, and with what evidence. Supply it, so the proposals build on it rather than rediscovering it.
- That you are asking for candidates and will choose among them yourself.
- A ceiling on the number of proposals. An unbounded request returns an unbounded list that nobody triages.

Prompt C.8. Enumerate candidate views before modeling

```
Decision this feeds: [DECISION]
Tables and fields available, with grain: [PASTE SCHEMA]
What has already been established, with its source:
[CERTIFIED FINDINGS, EACH WITH THE SECTION OR LOG ENTRY IT CAME FROM]

Propose at most [SIX] tables or views worth building before any model is
fitted. For each, state:
  the question it answers, in one sentence
  the fields required
  the output grain — one row equals one what
  the pattern you would expect if the established findings hold
  one anomaly that would be worth investigating
  one verification check the output must pass
```

```
CONSTRAINTS
Treat every proposal as a candidate for my specification, not as analysis.
Do not state findings. Do not rank the proposals by how interesting you
expect the answer to be.
Name any proposal that requires a field I have not listed, and mark it
UNSPECIFIED rather than assuming the field exists.
Make the proposals differ from one another in question, not only in form.
```

Likely failure modes.

- Proposals that require fields the schema does not contain, offered without comment.
- Proposals ranked by how interesting the assistant expects the answer to be, which is a finding stated before any evidence exists.
- Findings smuggled into the framing — "a view showing that the new stores are outperforming" rather than a view that would answer whether they are.
- Six proposals that are one proposal cut six ways, because nothing in the prompt asked for them to differ.

Audit after the response.

- Delete every proposal whose output grain you cannot state.
- Delete every proposal that requires a field the schema does not carry, or obtain the field first.
- Add each surviving proposal to the specification as a question, not as a planned finding.
- Check that the expected patterns are consistent with the certified findings you supplied; where they are not, that is a question worth asking before building anything.

What to record in Appendix D. Beyond the standard record of Section C.3: the schema and established findings you supplied, the proposals returned, which you kept, which you discarded and why.

C.7 Segmentation, Drivers, and Prediction Templates

From Chapter 6 onward the assistant can draft the entire analytic act, not merely its mechanics. Prompted with a customer file, a current assistant will build features, scale them, choose a number of clusters, fit the model, name the segments, and write the strategy narrative in one response. The four templates in this section are all two-prompt templates for that reason, and the audit between the prompts is the part that cannot be delegated.

Template C.9 — Build and Validate a Segmentation

Owned by. Chapter 6, Section 6.12. The first instruction block is that section's, verbatim; the second renders its prose requirements as a prompt.

Use this template when. You are building a customer-grain feature table and fitting a clustering, and you want the mechanics drafted without the conclusions attached.

What you must decide first.

- The feature list with derivation rules, in writing, before the first prompt. Feature derivation is safe to delegate only when the rules come from your definitions rather than the assistant's defaults.
- The population rule, the analysis date, and the window.
- The number of clusters — after the sweep, and by the braid of elbow, silhouette, and business judgment. Not in the first prompt.
- The withheld attributes you will profile on. They must be withheld from the fit, which means naming them before it.

Prompt C.9. Build the feature table and run the evaluation sweep

```
FEATURE LIST WITH DERIVATION RULES:
[PASTE]
POPULATION RULE: [E.G. CUSTOMERS WITH AT LEAST ONE ORDER IN THE WINDOW]
ANALYSIS DATE: [DATE]        WINDOW: [N] MONTHS
CERTIFIED TOTALS: revenue [$N], distinct orders [N], distinct purchasers [N]
TREATMENT CAPACITY OF THE PLATFORM: [N DISTINCT TREATMENTS]
```

```
Build [FEATURE TABLE NAME] per these rules; print the three reconciliation
figures; fit k-means for k = 2 through 8 on standardized inputs with a fixed
random seed; report inertias and mean silhouettes; recommend nothing yet.

Use only the derivation rules above. Do not add, drop, or transform a
feature I have not specified. Do not name or describe anything.
```

Prompt C.10. Fit the chosen solution and return the validation artifacts

```
I have chosen k = [K], by elbow, silhouette, and business judgment.
Do not revisit that choice.

Return, in this order and nothing else:
  the fit on standardized inputs with seed [SEED]
  the centroid table in standardized units and in native units
  adjusted Rand indices across five independently seeded fits
  adjusted Rand indices across two 80 percent subsamples
  a profile of each cluster on these withheld attributes: [LIST]

Write code that leaves the fitted scaler and the fitted model in clearly
named objects, and print the object names. Do not return the fit as prose
thresholds; an eyeballed threshold approximates a model without
reproducing it.
Use only the features and withheld attributes listed above.
Do not name the clusters. Do not describe them. Do not interpret the
profile. Do not refit at assignment time.
```

Likely failure modes.

- The confident k, presented as found rather than chosen, with none of the braid.
- Skipped scaling, which produces spend bands wearing segment costume and is invisible in the output.
- The invented persona — incomes, life stages, attitudes — asserted from a schema that contains none of them.
- The single-run solution, in which initialization luck ships as structure because nothing refit, perturbed, or compared.
- The narrated geometry, attaching motives to what is so far a partition of a feature table.
- The overfit taxonomy: asked for detailed segments, an assistant supplies nine.

Audit after the response.

- Verify one centroid by hand: filter the feature table to one cluster, average one feature, compare.
- Read the stability indices and reject the solution if the partitions do not reproduce across seeds and subsamples.
- Check the holdout profile against the predictions you wrote down before running it.
- Strike every noun in every segment name that no input feature or holdout column can redeem.
- Confirm the code leaves the fitted scaler and model in named objects you can reuse at assignment time. Assignment applies the fitted objects; it must never refit, and it must never approximate them with thresholds.

What to record in Appendix D. Beyond the standard record of Section C.3: both prompts, the reconciliation figures, the sweep, the stability indices, the holdout profile against your written predictions, every name you rejected and the evidence that would have been needed to keep it.

Template C.10 — Fit and Audit a Regression

Owned by. Chapter 7, Section 7.11. Both instruction blocks are that section's, verbatim; the frames around them are generalized with placeholders.

Use this template when. You have a written model specification — variables, reference category, formula — and want it fitted and reported without narration.

What you must decide first.

- The variable list, from your specification rather than the assistant's initiative.
- The reference category, explicitly. It determines what every dummy coefficient means.
- The predicted sign of every coefficient, written before the fit. The four-point audit compares against these.
- The covariance estimator. HC3 is the guide's default and the prompt states it so the assistant cannot inherit a nonrobust one.

Prompt C.11. Fit the model exactly as specified

```
VARIABLE LIST WITH DERIVATION RULES:
[PASTE]
DECLARED RESPONSE: [VARIABLE, IN ITS UNITS]
REFERENCE CATEGORY: [LEVEL]
COVARIANCE ESTIMATOR: HC3
MODEL FORMULA, exactly as written in my specification:
[FORMULA]

Fit this model as specified with HC3 standard errors; report the coefficient
table with 95 percent intervals in native units, the residual-versus-fitted
plot, variance inflation factors, and R squared; change nothing in the
specification; narrate nothing yet.
```

Prompt C.12. Draft the reporting sentences after the model survives

```
Here is the audited coefficient table:
[PASTE ONLY THE AUDITED NUMBERS]
Held constant in this model: [LIST]

Draft three sentences reporting these estimates as associations, using
"is associated with" or "predicts," naming units and the held-constant
list, and making no causal claim.

Do not add a recommendation. Do not use a number that is not above.
```

Likely failure modes.

- The confident narration — drivers returned as causes, because fluent causal prose saturates the training data. The arithmetic may be flawless while the verb is unearned.
- The silent specification: variables entered, dropped, reference category set, interaction included or not, none of it reported as a choice.
- The plausible-but-impossible coefficient, delivered with the same serene formatting as a correct one.
- The units slip — a coefficient narrated without units, or coefficients on different scales compared as though they were comparable.
- The diagnostics skip: no residual plot, because the default workflow fits and reports but does not examine.
- A main effect quoted unconditionally while its interaction sits in the same model.

Audit after the response.

- Run the four points of Table 7.5 in order: signs against your written predictions and the certified descriptives; magnitudes against hand-computable anchors; every coefficient restated aloud in its units, with conditional coefficients read conditionally; residuals inspected and the covariance estimator matched to what they show.
- Audit the second prompt's verbs as strictly as the numbers. Assistants reintroduce causal language at the narration step even when the numbers are clean.

What to record in Appendix D. Beyond the standard record of Section C.3: both prompts, the coefficient table, your predicted signs beside the fitted ones, which of the four points caught the most serious problem, every verb you changed in the drafted sentences.

Template C.11 — Draft a Predictive-Model Pipeline

Owned by. Chapter 8, Section 8.11. The instruction blocks preserve that section's wording; the frame is Table 8.1's seven declarations, with its *calibration metrics* row renamed **error metrics** — MAE and RMSE measure prediction error, and Chapter 9 uses *calibration* in its probabilistic sense.

Use this template when. You have declared the frame and the label and want the pipeline built. This is the delegation with the highest ratio of mechanical work to judgment, and also the one where a failure is least visible: a leaky pipeline does not look different from an honest one in a chat window — it looks better.

What you must decide first.

- All seven declarations of the frame, in writing, before the prompt: unit of prediction, target, features, horizon, decision metric, error metrics, baselines.
- The label with its four dates — feature-window start, snapshot, outcome-window start and end — and its eligibility rule.
- The split and cross-validation protocol with its seed.
- The range check that would convict a leak, computed from your snapshot before you see any result.

Prompt C.13. Build the pipeline exactly as framed

```
THE FRAME
  Unit of prediction   [ONE PREDICTION PER ...]
  Target               [QUANTITY, UNITS, OUTCOME WINDOW]
  Features             [FEATURE LIST WITH DERIVATION RULES]
  Horizon              [N MONTHS]
  Decision metric      [CAPTURE AT CAPACITY OR OTHER], against [INCUMBENT]
  Error metrics        MAE leading, RMSE beside it, both in [UNITS]
  Baselines            [MEAN], and [LAST-VALUE OR THE INCUMBENT LIST]

THE LABEL
  Snapshot date [DATE]; feature window [START] to [SNAPSHOT];
  outcome window [START] to [END]; eligibility rule [RULE]

THE PROTOCOL
  Split at the unit of prediction with seed [N]; [K]-fold cross-validation
  on the training rows only; every learned step inside a pipeline.

Build this pipeline exactly as specified; select among the candidates using
[K]-fold cross-validation on the training [UNITS] only; report one
leaderboard with [DECISION METRIC], MAE, and RMSE for every candidate and
both baselines; do not touch the test set; change nothing in the
specification; narrate nothing yet.

Before the code, list every feature and the window it is computed from, and
name any feature you cannot trace to the declared feature window.
```

Prompt C.14. Open the sealed test set once

```
I have selected [FINALIST] from the training-only leaderboard, using the
audit above. Do not revisit that choice.

Fit that exact pipeline on all training rows, open the sealed test set
once, and report [DECISION METRIC], MAE, and RMSE for the finalist and for
both baselines, on the same test rows.

Change no feature, transformation, setting, or evaluation rule. Do not tune
```

```
anything against what you see. Use only the specification above.
Narrate nothing.
```

Prompt C.15. Draft the comparison paragraph after the pipeline survives

```
Here are the audited results:
[PASTE ONLY THE AUDITED NUMBERS, INCLUDING BOTH BASELINES]

Draft the comparison paragraph for a marketing VP: the model's capture at
the program's capacity against the incumbent list, its test MAE against
both baselines in dollars, and no claim that any predicted difference is
caused by anything.

Use only the numbers above. Do not add a recommendation.
```

Likely failure modes.

- The convenient-table leak: features built from whatever table was supplied, including columns computed across the outcome window, because nothing in the prompt drew the snapshot wall. This is the single most common defect in AI-drafted pipelines (Kaufman et al., 2012).
- The celebration of the impossible — a leaky pipeline reporting a superb fit statistic, narrated as excellent.
- The missing opponent: metrics quoted with no baseline anywhere.
- The metric mismatch — distance metrics reported for a ranked-list decision, with no capture figure and no head-to-head against the incumbent.
- The vocabulary shuffle, in which classification words are applied to a numeric target.
- Hygiene slips: preprocessing fitted on pooled data, the test set consulted in a tuning loop, complexity chosen by watching test error.
- The deletion helpfulness — zero-value labels cleaned away as missing data, deleting the rows the model most needs.

Audit after the response.

- The test set is opened once, after the finalist is named and the training-only audit has passed. That is what the second prompt is for, and it is the step students most often collapse into the first.
- Run the five points of Table 8.5 in order: confirm the frame came back unedited; trace three features to their windows and run the range check; confirm every learned step sits inside a pipeline and the test set is untouched; confirm both baselines are present on every metric and the margin is computed; read the result for plausibility.
- If anything looks too good, stop and trace. In predictive work suspicion scales with success.
- Audit the second prompt's verbs: prediction language earns *expected* and *predicted*, never *will drive*.

What to record in Appendix D. Beyond the standard record of Section C.3: all three prompts, the training-only leaderboard, the feature-window trace, the range check with its result, the single sealed-test result with both baselines and the margin, and which audit point caught the most serious problem.

Template C.12 — Build and Audit a Classification or Churn Model

Owned by. Chapter 9, Section 9.11. The instruction blocks preserve that section's wording and controls, with the leaderboard separated into a ranking exhibit and a policy comparison so that the incumbent rule is judged at its operating point rather than ranked by AUC. The supplement is Table 9.5.

Use this template when. You are building or reviewing a classifier and need it graded on ranking evidence before any dollar figure is attached. Classification is where an assistant's unexamined defaults stop being statistical choices and start being business policy: a churn definition, a threshold, and a headline metric, all chosen silently in one fluent response.

What you must decide first.

- The churn or outcome label, verbatim: snapshot, both windows, activity rule, both eligibility conditions. The definition is supplied, never delegated.

- Which baselines are ranked and which are policies. A constant-score floor and a single-feature logistic model emit scores and belong on the ranking leaderboard; an incumbent yes/no rule is an operating point and belongs in the policy comparison, judged on its matrix and its economics.
- The threshold, derived from your program's own false-positive and false-negative economics. Not by the assistant, and not 0.5 by inheritance.
- The baselines, including the incumbent rule evaluated as a fixed policy rather than scored.
- The base rate, so that accuracy can be read against the do-nothing rule.

Prompt C.16. Build the classifier and grade it on ranking evidence only

```
THE LABEL, verbatim from my specification:
[SNAPSHOT, FEATURE WINDOW, OUTCOME WINDOW, ACTIVITY RULE, BOTH ELIGIBILITY
CONDITIONS]

FEATURES with derivation rules, computed from the feature window only:
[PASTE]

PROTOCOL: split with seed [N], stratified; stratified [K]-fold
cross-validation inside the training [UNITS].
BASELINES: a constant-score floor and a [SINGLE-FEATURE] logistic baseline
for the ranking comparison; [THE INCUMBENT RULE] evaluated as a fixed policy
at its own operating point, not scored.
COST ASSUMPTIONS, declared by me before any result: treatment cost [ ];
save rate [ ]; value of a save [ ].
THRESHOLD, derived by me from those costs: [P]

Build exactly this pipeline, and return two separate exhibits.

1  RANKING LEADERBOARD. The score-producing candidates and the
   continuous-score baselines only, selected and compared using stratified
   [K]-fold cross-validation on the training [UNITS], graded on
   cross-validated AUC alone. No dollar figures beside raw scores. Do not
   place [THE INCUMBENT RULE] on this leaderboard: it emits a yes/no
   decision, not a score, and AUC is not the instrument for it.

2  POLICY COMPARISON, after calibrating only the selected model inside the
   training data. Report its out-of-fold reliability table, then compare the
   calibrated model at the [P] threshold against [THE INCUMBENT RULE] as a
   fixed operating point, using the four confusion-matrix counts,
   precision, recall, treated-list size, and cross-validated expected net
   value under the cost assumptions above. Report precision as undefined,
   not zero, for any policy that treats nobody. Add a decile table and
   cumulative gains through the top two deciles for the selected calibrated
   model and for [THE INCUMBENT RULE].

Do not touch the test set. Do not output individual-level verdicts.
Choose no threshold yourself. Narrate nothing yet.
Use only the features listed above. Name any feature you cannot trace to
the declared feature window.
```

Prompt C.17. Draft the recommendation paragraph after the classifier survives

```
Here are the audited results:
[PASTE ONLY THE AUDITED NUMBERS]
Cost assumptions: [TREATMENT COST], [SAVE RATE], [VALUE OF A SAVE]

Draft the recommendation paragraph for a marketing VP: the model's capture
in the top two deciles against [THE INCUMBENT RULE]'s, the treated-list
size at the [P] threshold with the cost assumptions stated, and no sentence
```

```
that asserts an individual customer will churn.

Use only the numbers and assumptions above.
```

Likely failure modes.

- The accuracy celebration — a headline accuracy quoted without the base rate or the majority-class baseline, leaving the margin over doing nothing uncomputed.
- The silent threshold: a confusion matrix cut at 0.5 with no costs discussed, and every downstream number inheriting the default.
- The label improvisation — a churn definition invented and computed at the analysis date, building the label out of the feature window and manufacturing outcome-window contamination in one step. The resulting ranking is spectacular and spectacularly meaningless.
- The test-set race, in which a depth or a neighbor count is chosen by watching test numbers move.
- The mixed leaderboard, in which a fixed yes/no rule is given an AUC and ranked beside score-producing models. The number is computable and answers nothing: a rule with one operating point has no ranking to grade.
- The rebalancing reflex, which distorts the probability scale the threshold arithmetic consumes.
- The uncalibrated economics — expected-value tables computed from raw scores with no reliability exhibit anywhere.
- The verdict language, in which scores are narrated as facts about named individuals.

Audit after the response.

- Confirm the two exhibits came back separate, and that no dollar figure sits beside a raw score.
- Confirm the decile and gains exhibits cover both the model and the incumbent rule, so the reporting claim about the top two deciles has something behind it.
- Run Table 8.5's five points, then Table 9.5's four: base rate and floor stated with the margin computed; the confusion matrix's four counts re-added by hand and reconciled to the evaluation set, with precision and recall recomputed from raw counts; the threshold identified, its cost assumption stated, and the cut re-derived from the program's economics; calibration and selection hygiene confirmed, including that the reliability exhibit describes the model that was actually selected.
- Open the test set once, apply the threshold you derived, and re-add the matrix by hand.
- Audit the second prompt's verbs: propensity language earns *is likely to*, never *will*.

What to record in Appendix D. Beyond the standard record of Section C.3: both prompts, the ranking leaderboard and the policy comparison as two exhibits, the reliability table, the decile and gains exhibits, the hand-recomputed matrix, the threshold with its cost derivation and sensitivity, and which supplement point caught the most serious problem.

C.8 Forecasting and Experiment Templates

Two tasks in this guide have a property the others do not: the standard tooling's defaults are wrong for them. A forecasting pipeline built without instruction will shuffle ordered days into a random split, and an experiment analysis built without instruction will apply trial machinery to observational data. Neither error announces itself, and both produce excellent numbers. The templates in this section spend most of their length closing those two routes.

Template C.13 — Draft or Audit a Forecasting Workflow

Owned by. Chapter 10, Section 10.11. Both instruction blocks preserve that section's wording, generalized with placeholders. The supplement is Table 10.5.

Use this template when. You are building a forecast that a plan will consume. The prompt is long because each of its lines closes a specific default, and a shorter prompt gets the default.

What you must decide first.

- The three windows and their dates: rolling-origin selection, sealed holdout, and planning horizon. The holdout must resemble the decision, not merely be recent.
- The seasonal structure to fit, and the structure not to fit. Annual seasonality is never fitted from two repetitions.
- The known calendar — openings, promotions, and holidays in every window, with dates. No period-7 method can find an event nobody mentioned.
- The predicted rolling-origin error for every row of the leaderboard, computed by hand for the baselines, written before the pipeline runs.

Prompt C.18. Build the forecast with the escape routes closed

```
SERIES      [DAILY / WEEKLY] [QUANTITY] from [CERTIFIED TABLE], reconciled
            to [TOTAL]
WINDOWS     rolling-origin selection through [DATE];
            sealed holdout [START] to [END];
            planning horizon [START] to [END]
METRIC      MAPE with denominators asserted and the minimum printed,
            MAE reported beside it
OPPONENTS   seasonal-naive at period [P]; naive as the floor
SEASONALITY fit [WEEKLY ONLY]; no annual fitting, under any circumstances
CALENDAR    [OPENINGS, PROMOTIONS, AND HOLIDAYS IN EVERY WINDOW, WITH DATES]

Compare candidates on rolling origins inside the selection window only; do
not touch the sealed holdout until I name the method; report every candidate
and both baselines in one table, by horizon week, with the number of errors
behind each figure; produce [80] percent intervals and aggregate paths
before quantiling for any weekly or total figure; narrate nothing yet.

Do not shuffle. Do not use a centered window anywhere in the pipeline.
Do not substitute a method I have not named.
```

Prompt C.19. Open the sealed holdout once, then produce the planning forecast

```
I have selected [METHOD] from the rolling-origin comparison. Do not revisit
that choice.

Fit it on the complete selection window, open the sealed holdout once, and
report holdout MAPE and MAE beside both baselines on the same dates.

Then refit through [FINAL OBSERVED DATE] and produce the planning forecast
and its [80] percent intervals under the named scenarios, aggregating paths
before quantiling for any weekly or total figure.

Use only the specification and calendar above. Do not reconsider the
method. Do not report the planning forecast as a graded one; it has no
actuals. Narrate nothing.
```

Prompt C.20. Draft the plan paragraph after the forecast survives

```
Here are the audited results and the override log:
[PASTE ONLY THE AUDITED NUMBERS AND THE SIX-ELEMENT OVERRIDE ENTRIES]

Draft the plan paragraph: the [MONTH] range under each named scenario, the
margin over seasonal-naive in dollars, the overrides listed with owners and
review dates.

Every figure must appear above. Do not state a point without its range.
```

Likely failure modes.

- The shuffled split: random or k-fold evaluation applied to ordered data, which is leakage by design and is caught in one line by the date-order check.
- The reused holdout, in which four methods are compared on the block later reported as the grade.
- The mismatched season — a holdout chosen for convenience, so a holiday plan is validated on late spring.
- The invented season: a period guessed at 12 on daily data, or annual seasonality fitted from two cycles.
- The silent extrapolation, straight through an announced opening and straight through December.
- The benched-metric revival — MAPE quoted on a series with near-zero days.
- The summed interval, in which weekly and season-total bounds are produced by adding daily bounds together. The numbers look plausible, which is why this one survives review.
- The naked point: a forecast delivered without an interval.
- The method upgrade, in which the assistant reaches for machinery the analyst cannot explain to the meeting that funds it.

Audit after the response.

- The holdout is opened once, after the method is named, and the planning forecast is refitted separately and never called a graded one. The second prompt exists to keep those three acts visibly distinct.
- Run Table 8.5's five points, then Table 10.5's four: the date-order check at every origin, with no centered window anywhere; selection separated from the grade; the calendar and the breaks declared and matched to overrides or stated assumptions; metric and interval honesty, including asserted MAPE denominators and path-first aggregation for any weekly or total figure.
- Compare the headline against the band you predicted before it ran, and classify any excess as pipeline rather than prophecy until it is traced.
- Audit the second prompt's verbs: a forecast is *expected*, *projected*, *likely between*; it is never *will be*.

What to record in Appendix D. Beyond the standard record of Section C.3: all three prompts, your predicted leaderboard band beside the actual, the date-order check, the single sealed-holdout result, every override with its owner and review date, and which supplement point caught the most serious problem.

Template C.14 — Draft an Experiment Registration

Owned by. Chapter 11, Section 11.11. The instruction block is that section's, verbatim, generalized with placeholders. The fifteen rows are Table 11.3's.

Use this template when. You are designing a test and want the registration drafted before any data exists. The chronology is the control: this template is only valid if it runs first (Nosek et al., 2018).

What you must decide first.

- What you are testing, stated as a treatment applied to an eligible population and measured on one primary metric.
- The columns you actually have. A guardrail that cannot be computed from them is a design defect, not a missing nicety.
- That you want gaps made visible. The final clause of the prompt is the whole prompt.

Prompt C.21. Draft a preregistration with the gaps made visible

```
I am testing whether [TREATMENT] lifts [PRIMARY METRIC] among [ELIGIBLE
POPULATION]. The columns available to me are: [PASTE SCHEMA].

Draft a preregistration covering every row of the following template —
decision, primary metric with its four elements, arms including what control
receives, treatment dose in number of exposures, assignment mechanism, unit
of randomization, eligible population, sample size with the power
calculation shown, duration, planned analysis, decision rule, stopping rule,
secondary metrics labeled exploratory, and guardrails that are computable
from the columns I have listed.
```

```
Where the information I gave you is insufficient to fill a row, write
UNSPECIFIED and ask me; do not fill it yourself.

For every guardrail you propose, name the field it would be computed from.
Do not propose a metric, a dose, or a duration I have not supplied.
```

Likely failure modes.

- The invented design. Asked to design a test, an assistant fills every gap the prompt left — inventing a metric definition, choosing a duration, assuming a unit of randomization, assuming a treatment dose, sometimes assuming an eligibility rule — silently and plausibly, so that the registration reads like a plan you made.
- A treatment dose left implicit, which leaves the break-even lift undefined and therefore the decision rule meaningless.
- Guardrails that name a harm no available column can measure.
- A stopping rule phrased as guidance rather than as a rule.

Audit after the response.

- Count the UNSPECIFIED rows. That count measures your brief, not the assistant.
- Check specifically whether the dose and the guardrail-computability rows were left unspecified or invented.
- Confirm every declared guardrail names the field it would be computed from.
- Confirm the decision rule maps a statistical output to an action, and that the stopping rule fixes the number of looks.
- Date the registration and freeze it. A registration edited after data exist is not one.

What to record in Appendix D. Beyond the standard record of Section C.3: the prompt, the draft, the count of UNSPECIFIED rows, whether the dose was invented or left open, what that count revealed about your own brief.

Template C.15 — Analyze and Stress-Test an Experiment

Owned by. Chapter 11, Section 11.11. Both instruction blocks preserve that section's wording, generalized with placeholders. The supplement is Table 11.7.

Use this template when. The outcome window has closed and you are analyzing exactly what you registered. The registration is pasted back in, because the analysis is graded against it rather than against what now looks interesting (Kohavi et al., 2020).

What you must decide first.

- The interval width you expect, computed from the arm sizes and the baseline rate. An interval far narrower than predicted means the analysis grain slipped.
- The break-even lift, computed in advance from the declared dose. An analyst who selects the contribution figure after seeing the estimate has invented a new way to peek.
- That the primary comparison is intention-to-treat, and that operational losses are reported rather than filtered.
- What the registration says about missing outcomes. A delivery failure is not grounds for removing anyone; an unobserved outcome is a different problem, and it cannot simply sit in the denominator. If the registration is silent, that is the finding, and the analysis stops until it is resolved.

Prompt C.22. Analyze exactly as registered

```
Here is the registration.

[PASTE THE DATED REGISTRATION]

Analyze the attached file exactly as registered — intention-to-treat, no
other metrics, no subgroups, one analysis. Report the assignment-flow
counts, the control rate, the treatment rate, absolute lift in percentage
points with a 95 percent interval on the difference, relative lift with a 95
percent interval on the ratio computed as a ratio, and the p-value.
Do not interpret.
```

```
Do not remove any assigned unit from the assignment-flow table, and do not
redefine the analysis population. Analyze missing outcomes according to the
preregistered rule. If the registration contains no such rule, report the
count by arm, mark the analysis decision UNSPECIFIED, and stop before
computing the effect.
```

Prompt C.23. Argue against your own result

```
Here is the audited estimate and its interval: [PASTE]
The test ran on [POPULATION], over [WINDOW], at a dose of [DOSE].
The rollout under consideration is [SCOPE, DURATION, DOSE].

List every reason this estimate might not transfer to that rollout, and
every contamination path you can construct.

Use only what I have described above; do not assume a fact about the
population, the dose, or the program that I have not stated.
Do not rank the reasons. Do not reassure me. Do not rewrite the estimate.
```

Likely failure modes.

- The observational file analyzed as a trial — the highest-frequency and highest-cost failure, invited by the request itself whenever the prompt says *campaign* instead of *experiment*.
- The uncomplaining test on peeked data. The assistant computes significance for whatever window it is handed and will not ask how many times the data have been examined.
- The per-protocol drift, in which undelivered customers are dropped and the randomized comparison quietly becomes an observational one.
- The segment harvest, testing twenty subgroups and reporting the significant ones at machine speed.
- The lift ambiguity: relative lift quoted without its baseline, or a ratio interval produced by dividing a difference interval by the control rate.
- The naked winner, with no interval, no practical-significance line, no stated dose, and no statement of the tested population.
- The grain slip, analyzing orders or sessions after customers were randomized.
- The power omission, narrating a null as *no effect* rather than as an effect the test could not have detected.

Audit after the response.

- Run Table 8.5's five points, then Table 11.7's three: registration precedes analysis, with every deviation listed as a deviation; assignment integrity, with the flow record reported from randomization through outcome observation and the analysis grain equal to the assignment grain; inference honesty, with absolute lift beside its baseline, a difference interval and a separately constructed ratio interval, and the whole interval compared against the break-even line rather than the point estimate.
- Grade the interval width against your prediction.
- Harvest the adversarial prompt's output into the deliverable's limitations, discarding items with one sentence of justification each.

What to record in Appendix D. Beyond the standard record of Section C.3: both prompts, the assignment-flow counts, the two intervals, the break-even comparison across the whole interval, the adversarial list with the items you kept and discarded.

C.9 Visualization and Communication Templates

Part III changes the shape of the problem. A chart's mechanics and its meaning are produced by the same act, and a sentence about a view is the medium in which the claim is made. There is no later stage at which the meaning gets audited unless the analyst inserts one. The three templates in this section insert it, and all three run the five-step routine of Section C.1.1 rather than the two-prompt pattern.

All three templates repair against Appendix E, which holds the visualization checklist in runnable form. Where an entry says *repair against Appendix E*, the checklist is the instrument and this appendix only routes you to it.

Template C.16 — Ask for Candidate Visual Specifications

Owned by. Chapter 12, Section 12.12. The first prompt preserves that section's wording and adds the candidate framing Table C.2 requires; the second consolidates the palette, footer, and alternative-text uses that Sections 12.12 and 13.11 both name.

Use this template when. You have a question and want a slate of candidate views specified rather than rendered. Specifications can be audited; rendered charts are persuasive in a way their specifications are not, and the human tendency is to evaluate whether a chart is attractive rather than whether it is right.

What you must decide first.

- That every element the assistant returns — the mark, the encodings, the aggregation, the axis ranges, the sort, the palette — is a candidate for your approval and not a decision. Table C.2's last row governs: the analyst sets the axis. A proposed range that does not begin at zero on a length-encoded mark is a defect to catch, not an instruction to follow.
- The schema, pasted. Almost every invented-field failure is a prompt that did not include it.
- The grain of the source table, stated. This is the line that prevents a line-grain average being presented as a per-order quantity.
- The question, in one sentence, as a comparison the reader must make.
- Your prediction, written from the specifications alone: for each candidate, the comparison, the channel carrying it, that channel's rank in the perceptual hierarchy, and the two faults you expect.

Prompt C.24. Request candidate view specifications

```
I have [TABLE] at [GRAIN], with these columns [PASTE THE SCHEMA]. I want to
answer this question: [QUESTION]. Propose three candidate views. For each,
state the mark type, every field-to-channel encoding, the aggregation and
the grain one mark represents, the axis ranges, the sort order, and the
palette. Do not use any column not in the schema I pasted; if the question
requires a field I have not given you, say so instead of assuming it. Do not
render or describe conclusions.

Treat every element as a proposal for my approval. For any axis range that
does not begin at zero, state the reason in one line; do not choose a range
to make a difference look larger.
```

Prompt C.25. Request the palette, the footer, and the alternative text

```
For the view specified as [PASTE THE CHOSEN SPECIFICATION], produce three
things and nothing else.

1  A color-blind-safe [CATEGORICAL | SEQUENTIAL | DIVERGING] palette as an
   explicit list of hex values, with a note on how each pair reads in
   grayscale.
2  The footer line, using only the values below: source table, date range,
   filters applied, grain of aggregation, and extract refresh date.
   [SUPPLY THE FIVE VALUES]
3  Alternative text giving the view's purpose, population, units, the
   major pattern, and any material exception — without adding a causal,
   predictive, or evaluative claim the view does not support.

Do not use a red-to-green palette to carry an ordered field.
Do not add a value I have not supplied.
```

Likely failure modes.

- The truncated axis to order, produced whenever the request contains a phrase like *make the difference clear*.

- The grain slip — a per-order metric computed from line-grain rows, which is a correct average of the wrong thing.
- The default pie for a composition question.
- The palette mismatch in both directions, with an accessibility omission alongside it, since nothing in the request will have mentioned color-vision deficiency.
- The invented field, referencing a column that would be natural for the question and does not exist.
- The wrong-purpose chart — a trend answer to a comparison question, which will look entirely reasonable.
- The undisclosed filter, which leaves no trace in the image.
- The default sort, left in place constantly.
- The overclaiming title, containing a verb the view has not earned.

Audit after the response.

- Build the specification exactly as written, including the faults. A specification corrected silently during construction cannot be audited.
- Run the three transferable points of Table 8.5 and all three of Table 12.5, line by line, on the built view.
- Compare the faults you found against the faults you predicted, and record both the misses and the false alarms. A fault you predicted and did not find is as informative as one you missed.
- Repair against Appendix E, one mechanic at a time, with a one-sentence note per repair recording what changed in the *reading* rather than what changed in the chart.

What to record in Appendix D. Beyond the standard record of Section C.3: the request prompt with the schema you pasted, the candidate specifications, any column the assistant invented despite the schema, your written predictions, the audit, your grade of your own predictions including false alarms, the repairs with their reading notes.

Template C.17 — Audit a Chart or Tableau View

Owned by. Chapter 12, Section 12.12, adversarial use. The checklist it applies is Appendix E.

Use this template when. A view exists — yours, a colleague's, or one an assistant drafted — and you want every way it could mislead a reader enumerated before it circulates. Handed a description of a finished view and asked what could go wrong, an assistant produces a checklist that will often contain items the builder missed.

What you must decide first.

- That you will describe the view completely, including the filter state and the footer. An audit of a partial description audits a partial chart.
- What the reader is supposed to conclude. The audit is about the gap between that and what the view supports.
- Your own one-minute check first: read the axis floor, name the grain, name the comparison. Run it before you ask, so the response is graded rather than trusted.

Prompt C.26. Enumerate how a finished view could mislead

```
Here is a complete description of a finished view.
  Mark type: [ ]
  Every field-to-channel encoding: [ ]
  Aggregation, and the grain one mark represents: [ ]
  Axis ranges, both axes: [ ]
  Sort order: [ ]
  Palette and its type: [ ]
  Filter state, including defaults: [ ]
  Title, subtitle, and footer as written: [ ]
The reader is meant to conclude: [ONE SENTENCE]

List every way this view could mislead that reader. Organize the list under
these headings and nothing else:
  grain and aggregation
  scale and baseline
  encoding, and the comparison the reader must make
```

```
  color and accessibility
  filter state, and what is not shown
  the title's verb, and every number in it

For each item, state the change in what the reader would conclude, not the
change to the chart.
Use only what I described. Do not assume a field, a filter, or a value I did
not state. Do not propose a redesign. Do not comment on taste.
```

Likely failure modes.

- Objections about taste rather than about reading — font, color preference, placement — which are unfalsifiable and therefore unhelpful.
- Assumed filters, fields, or values you did not describe, producing defects the view does not have.
- A redesign offered in place of a defect list, which moves ownership of the encodings.
- Silence about the omissions, which are the hardest class: an undisclosed filter leaves no trace in an image and will not be surfaced unless the prompt asks for it by name.

Audit after the response.

- For every item, decide whether it is a defect, a judgment call, or noise. Appendix E distinguishes stop-ship failures from judgment calls; use its labels.
- Repair the stop-ship items against Appendix E, one mechanic at a time.
- For each repair, write one sentence on what changed in the reading. *A reader now sees a quarter that moved slightly, rather than a quarter that appeared to jump twelvefold* is the form.
- Confirm the footer carries source, window, filters, grain, and refresh date before the view leaves the workbook.

What to record in Appendix D. Beyond the standard record of Section C.3: the description you supplied, the defect list, which items you accepted and which you discarded with one sentence each, the repairs with their reading notes.

Template C.18 — Draft and Test Executive Communication

Owned by. Chapter 13, Section 13.11. The first instruction block preserves that section's wording, generalized with placeholders; the second renders its rewrite request as a template.

Use this template when. You have a verified view or a finished recommendation and need language for it. This is the narrowest delegation in the guide: the assistant may generate candidates, and the analyst must verify every one against the view it sits on, one sentence at a time, before any of them reaches a canvas.

What you must decide first.

- What the view actually shows, in its own units, including the values a reader can see. Supply nothing else — in particular, do not paste the chapter's conclusions, because the assistant will use them and the title will contain a number the view does not.
- The population the view covers, and its boundary. A statement about four mature stores must not become a statement about a format.
- Your prediction: how many of the candidates will overclaim, and in which of the three shapes — causal or predictive verb, generalized population, imported number.
- For the rewrite prompt, the reader: role, what she knows, and how long she will spend.

Prompt C.27. Generate candidate assertion titles with verification lines

```
Here is a description of one view from a dashboard: [DESCRIPTION IN THE
VIEW'S OWN UNITS, INCLUDING THE REFERENCE LINE, THE POPULATION IT COVERS,
AND THE VALUES A READER CAN SEE].

Write twelve candidate titles as complete sentences.

Do not use any number that I have not given you. Do not assert a cause, a
forecast, or a payback period. Do not generalize beyond the [N] [STORES /
```

```
CUSTOMERS / PERIODS / EXPERIMENTS] represented in the view.
For each candidate, state in a second line exactly which marks in the view a
reader would have to look at to verify it.
```

Prompt C.28. Rewrite for a stated reader without losing a finding

```
Rewrite the paragraph below for [READER: ROLE, WHAT SHE ALREADY KNOWS, HOW
LONG SHE WILL SPEND ON IT].

[PASTE YOUR PARAGRAPH]

Keep every interval, every threshold, every reversal condition, and every
qualification of population, window, and dose. If you shorten a sentence,
do not remove a clause that states a limit.

Use only what is in the paragraph above. Do not introduce a number, a
claim, or a qualification that is not already there.

Return the rewrite, and beneath it a complete list of every clause you
removed, merged, or reworded, with the original wording beside the new one.
```

Likely failure modes.

- The overclaiming title, in its three shapes. It is the modal failure, because a more interesting sentence is better writing and the assistant is optimizing for writing.
- The confident recommendation, which drops the interval, the threshold, and the reversal criterion while keeping the tone.
- The disappeared uncertainty. Asked to make a paragraph clearer, an assistant removes the interval before it removes anything else, because the interval is the clause that complicates the sentence.
- The generic audience, produced by any prompt that does not name the reader. Pasting the audience analysis of Section 13.2 into the rewrite prompt changes the output substantially and costs one paragraph.
- The invented finding — a connective sentence asserting a relationship between two views that no view establishes. It reads as synthesis; it is confabulation, and it is caught only by the rule that every sentence must be traceable to a specific view.

Audit after the response.

- Take each candidate, cover the description you supplied, look only at the built view, and decide whether a reader could confirm the sentence from the marks. Reject on the first failure rather than repairing, because repairing reintroduces your own knowledge into a sentence that must stand without it.
- Grade the overclaim count and shapes against your prediction, recording misses and false alarms.
- For the rewrite, diff the two versions clause by clause rather than reading the new one. The missing clause is invisible in a text that reads well.
- Show the surviving title on its view to someone who has seen neither, ask what it says, and confirm her paraphrase is the claim you intended rather than the stronger one next door. The reader's first interpretation is the usability evidence, not her opinion of the design (Krug, 2014).
- Run C1 through C3 of Table 13.6 on the finished artifact.

What to record in Appendix D. Beyond the standard record of Section C.3: both prompts, the twelve candidates with their verification lines, your predicted overclaim count and shapes beside the actual, the clause-by-clause diff with every removal classified as legitimate compression or deletion of a finding, the reader's verbatim paraphrase. Section 13.11 names two further failure modes that this entry deliberately does not carry, because neither prompt above can produce or repair them. The **layout without a hierarchy** is a dashboard defect, and its fix is to supply the KPI hierarchy of Section 13.4 as an input rather than to audit a sentence. The **accessibility and provenance omissions** are omissions rather than errors, and they are caught by the export and delivery section of Appendix E. Template C.19 picks up both at the deliverable level.

C.10 Review and Close

One template belongs to no single chapter. Four chapters name the same use — the assistant as an adversary — and each says it is where the tool earns most: Section 2.8 on the specification, Section 11.11 on a causal estimate, Section 12.12 on a finished view, Section 13.11 on a finished dashboard. Template C.19 generalizes them to a whole deliverable, and it is the last thing you run before you sign.

Template C.19 — Conduct an Adversarial Final Review

Owned by. Sections 2.8, 11.11, 12.12, and 13.11, which each name adversarial review as the highest-value delegation. The defect categories consolidate the audit tables of Section C.3.

Use this template when. The deliverable is finished and you are about to submit or circulate it. Run this before the last read-through, not instead of it.

What you must decide first.

- Whether the deliverable may leave your environment at all. This template asks you to paste a finished artifact, which is the largest single disclosure in the appendix. Run a manual or approved local privacy screen **first**. Do not use this template on identifiable, confidential, contractually restricted, or proprietary material unless the assistant operates inside an environment authorized for it. The final defect heading asks the assistant to find sensitive data; by then it has already been supplied, so that heading is a backstop and not the control.
- That you want a defect list and not a rewritten deliverable. A rewrite at this stage transfers authorship of the claims.
- What evidence stands behind each figure — design, window, population, method, and the interval or baseline. Without it the review can only check internal consistency.
- Which of the defects, if found, would stop submission. Deciding that in advance stops the list being triaged by how tired you are.

Prompt C.29. Adversarial review of a finished deliverable

```
Here is a finished deliverable: [PASTE THE MEMO, THE TITLES, THE TABLE OF
FIGURES, AND THE LIST OF VIEWS WITH THEIR GRAINS].

Here is the evidence behind it: [DESIGN, WINDOW, POPULATION, AND METHOD FOR
EACH CLAIM, AND THE INTERVAL OR BASELINE FOR EACH FIGURE].

Act as a skeptical reviewer who will be blamed if this is wrong. Produce a
numbered defect list under these headings:
  claims not supported by the evidence described
  a missing baseline or comparison
  a grain that changes between a figure and the sentence about it
  a metric used with two different definitions
  causal or predictive verbs the evidence does not license
  uncertainty stated nowhere
  a number in the prose that appears in no exhibit
  anything a reader could reasonably conclude that the evidence does not
    support
  sensitive or identifying data visible anywhere (a backstop; the privacy
    screen runs before this prompt, not inside it)
  missing AI-use documentation

For each defect, quote the sentence or name the exhibit.
Use only the deliverable and the evidence described above; do not assume
context I have not supplied.
Do not rewrite anything. Do not rank or score the deliverable.
Do not raise an objection that would apply equally to any deliverable.
End with the three questions you would ask the analyst before signing.
```

Likely failure modes.

- Generic review comments that would apply to any deliverable.
- Invented defects, produced by assuming context you did not supply.
- Silence on omissions, which is the category the review is most valuable for and least likely to volunteer.
- A rewritten version supplied helpfully alongside the list.

Audit after the response.

- Triage every item into defect, judgment call, or noise, and record one sentence for each discard.
- Repair the defects. Re-run the specific audit from Table C.4 that governs whatever you changed — a repair is a new artifact and inherits its own audit.
- Answer the three closing questions yourself, in writing. If you cannot, the deliverable is not finished.
- Confirm the four questions of Section 1.9 can be answered yes for every claim that survives.

What to record in Appendix D. Beyond the standard record of Section C.3: the deliverable version reviewed, the defect list, the triage with a sentence per discard, the repairs and the audits they triggered, your answers to the three closing questions.

C.11 Anti-Patterns

Every chapter from Chapter 4 onward prints a deliberately loose prompt and grades what comes back, because reproducing the failure is how the failure becomes legible. Those prompts are collected in Table C.6 with what each one leaves unspecified and the minimum addition that makes the task auditable. They are worth running once each: an exchange in which the assistant performed badly and you caught everything is a better artifact than one in which it performed well.

Table C.6

Weak requests, what they leave unspecified, and the minimum repair

The weak request	What is unspecified	Minimum repair
“Analyze this dataset and give me insights.”	The decision, the unit of analysis, the comparison, and what would count as an answer	Template C.1, then Template C.8. State the decision and the output grain; ask for candidate views, not findings
“Clean this file.” (Exercise 4.6)	Which step, which remedy, what may be dropped, and what the totals should be afterward	Template C.5. One step, one named remedy, predicted row count and revenue total before the code
“Summarize this sales data and tell me what it means.” (Exercise 5.6)	The output grain, every denominator, the measure of center, and the totals it must reconcile to	Template C.7. Add the certified totals and forbid causal verbs
“Segment these customers and describe the segments.” (Exercise 6.6)	The features and their derivation rules, the population, the number of clusters, whether inputs are scaled, and what the names may assert	Template C.9, in two prompts, with the validation artifacts required before any name is written
“Find what drives customer spend and tell me what to do about it.” (Exercise 7.6)	The specification, the reference category, the covariance estimator, and the verbs the evidence licenses	Template C.10. Supply the formula; forbid narration in the first prompt
“Build me the best possible model — maximize accuracy.” (Exercises 8.6, 9.6)	The frame, the label with its four dates, the snapshot wall, the baselines, and the decision metric	Templates C.11 and C.12. Seven declarations, the label verbatim, both baselines, and the test set untouched
“Forecast our daily revenue for the next eight weeks — maximize accuracy.” (Exercise 10.6)	The three windows, the seasonal structure, the calendar, the metric’s denominators, and the intervals	Template C.13. Selection separated from the grade; annual fitting forbidden; calendar supplied

The weak request	What is unspecified	Minimum repair
“Check whether the campaign worked, and see which customer segments responded best.” (Lab 11.2)	Whether the data are experimental, what was registered, how many looks have occurred, and whether the subgroups were chosen before or after	Templates C.14 and C.15. Register first; paste the registration back in; one analysis, no subgroups
“Make a chart showing how much the suburban stores have grown, and make the growth clear.” (Lab 12.2)	The schema, the grain, the axis range, and what “clear” is permitted to do to the axis	Template C.16. Paste the schema, state the grain, ask for a specification rather than a chart
“Make this recommendation more confident.” / “Make this punchier.”	Which clauses may be cut. The interval goes first, because it is the clause that complicates the sentence	Template C.18, second prompt. Require the removed-clause list and diff rather than read
“Fix the code.”	The error, the expected result, the data types, and what the cell is allowed to change	Template C.6. Paste the traceback and the expectation; forbid unrequested additions

The pattern across all eleven is the same. Each weak request delegates a decision the requester did not know she was delegating, and the assistant makes it silently because nothing in the request suggested it was open. The repair is never longer prompting for its own sake; it is naming the decision.

CONCEPT

Fluent Is Not Verified

An assistant-produced analytics failure often does not look like a failure. It looks like finished work: a table that adds up within itself, a model whose fit statistic is excellent, a chart that is well designed, a paragraph that reads better than the one you wrote. Presentation quality does not establish correctness, because presentation quality is what the tool optimizes.

So before any response leaves your desk, answer Section 1.9’s four questions in order. Does the output answer the question I specified? Does one row still represent what I said it represents? Can I reproduce at least one number by hand? Can I state in one sentence what this result does not show? A no anywhere is the finding.

Source: Course concept developed for this guide, informed by National Institute of Standards and Technology (2024) and Dell’Acqua et al. (2023).

C.12 Template Index and Exclusion Register

Table C.7 is the chapter-indexed prompt finder. It maps each chapter’s AI-assistant section and its AI in Practice box to the templates that generalize them, so that a student working through a chapter can find the template without reading this appendix from the front.

Table C.7

Templates by chapter

Chapter	AI-assistant section and box	Templates
1	Sections 1.7 and 1.9; “Review Before You Rely.” No prompt is printed; the box is the general four-question audit	The audit in Section C.3 applies to every template
2	2.8 AI as a Decision-Support Assistant; “Ask for the Other Side”	C.1, C.2
3	3.9 AI as a Measurement and Documentation Assistant; “Draft the Dictionary, Audit the Draft”	C.3, C.4
4	4.10 AI as a Data Preparation Assistant; “Delegate the Code, Never the Count”	C.5, C.6

Chapter	AI-assistant section and box	Templates
5	5.10 AI as a Descriptive Analytics Assistant; "The Summary That Must Add Up"	C.7, C.8
6	6.12 AI as a Segmentation Assistant; "The Cluster Solution That Must Survive a Refit"	C.9
7	7.11 AI as a Regression Assistant; "The Regression That Must Survive Its Audit"	C.10
8	8.11 AI as a Predictive-Modeling Assistant; "The Pipeline That Must Survive Its Audit"	C.11
9	9.11 AI as a Classification Assistant; "The Classifier That Must Survive Its Audit"	C.12
10	10.11 AI as a Forecasting Assistant; "The Forecast You Priced Before You Ran It"	C.13
11	11.11 AI as an Experimentation Assistant; "Preregister With the Assistant, Then Audit the Analysis Against the Registration"	C.14, C.15, C.19
12	12.12 AI as a Chart-Drafting Assistant; "Predict the Faults Before You Look"	C.16, C.17
13	13.11 AI as a Communication Assistant; "Draft the Titles, Then Prove Them"	C.18, C.19

Table C.8 records what this appendix does not contain and why. A library that silently omits a template looks identical to one that never considered it, and the reasons are worth stating because several of them are decisions rather than oversights.

Table C.8

Exclusion register

Not included	Reason
A template for Chapter 1	Chapter 1 has no AI-assistant section and prints no reusable prompt. Its contribution is the four-question audit of Section 1.9, which governs every template here, and the compact record of Table 1.6, which every entry invokes.
Separate templates for the four segmentation tasks the blueprint lists	Chapter 6 prints one specification prompt and states the second in prose. Publishing four templates would promise a division of labor the chapter does not teach. Template C.9 carries both prompts.
Separate templates for the four executive-communication artifacts	Chapter 13 prints one title prompt and one rewrite request. Template C.18 carries both; the memo structure and the audience-testing protocol belong to Sections 13.10 and 13.12 and are not prompts.
A template for error analysis	Sections 8.11 and 9.11 both print a one-line error-analysis question. Neither needs a template; both are folded into the audit steps of Templates C.11 and C.12.
Templates for alternative text and footer drafting	Sections 12.12 and 13.11 name both as good uses. They are one prompt between them and are folded into Template C.16.
Prompts that ask an assistant to interpret, recommend, or conclude	No chapter authorizes one, and eleven chapters explicitly forbid it at the point where it would occur.
A prompt for the deliberately loose requests	The eleven weak requests in Table C.6 are reproduced as anti-patterns to be run and graded, not as templates to be adapted.
Vendor-specific or model-specific phrasing	Every template is written to be tool-neutral. Where a chapter names a tool, it is because the course requires it, not because the prompt depends on it.
A rewritten "good" version of each weak prompt	The chapters ask students to author those rewrites (Exercises 4.6, 5.6, 6.6). Supplying them here would answer a graded exercise.

One dependency is recorded rather than excluded. Templates C.16 and C.17 repair against Appendix E, and Template C.17's triage uses its pass, caution, and stop labels. Chapter 12 invokes Appendix E eight times and

Chapter 13 once, so the dependency is the chapters' rather than this appendix's; but until Appendix E is published, those two entries point forward rather than across.

References

Dell'Acqua, F., McFowland, E., III, Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K., Rajendran, S., Krayer, L., Candelon, F., & Lakhani, K. R. (2023). *Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality* (Harvard Business School Technology & Operations Management Unit Working Paper No. 24-013). https://doi.org/10.2139/ssrn.4573321

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data, 6*(4), Article 15. https://doi.org/10.1145/2382577.2382579

Kohavi, R., Tang, D., & Xu, Y. (2020). *Trustworthy online controlled experiments: A practical guide to A/B testing.* Cambridge University Press. https://doi.org/10.1017/9781108653985

Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability* (3rd ed.). New Riders.

National Institute of Standards and Technology. (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile* (NIST AI 600-1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.600-1

Nosek, B. A., Ebersole, C. R., DeHaven, A. C., & Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences, 115*(11), 2600–2606. https://doi.org/10.1073/pnas.1708274114

Appendix D

AI-Use Documentation Template

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · August 2026
Except where otherwise noted, this appendix is licensed under CC BY 4.0.

Appendix Information

PURPOSE

This appendix is the standard record for AI-assisted work throughout the guide and the course. It exists so that an analysis produced with assistance can be audited afterward by someone who was not present: what was asked, what came back, what was predicted before it was run, what was kept, what was rejected, what check was performed, and what the analyst decided that the tool could not decide. It supplies two levels of record — a compact one that lives beside the work in a notebook, and a full one submitted with graded work — built on one field set that does not change between them.

VERSION AND DATE

Version 1.0 · August 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). AI-use documentation template. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Appendix D, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Except where otherwise noted, this appendix is licensed under a Creative Commons Attribution 4.0 International License.
Copyright © 2026 by Jose Mendoza.

Google Colab, Google Drive, Google Docs, and Google Sheets are products of Google LLC. Microsoft Word and Microsoft Excel are products of Microsoft Corporation. Tableau Desktop and Tableau Prep are products of Salesforce, Inc. ChatGPT is a product of OpenAI, Claude is a product of Anthropic, Gemini is a product of Google LLC, and GitHub Copilot is a product of GitHub, Inc. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this appendix, including outlining, preliminary drafts, revision of prose, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the appendix's scope, boundaries, field set, and content, and reviewed and approved all AI-assisted material: every field in the published record was traced to the chapter that requires it, and every numerical value in this appendix and its companion worked example was recomputed from the published arithmetic of the chapter that owns it and confirmed. Responsibility for the accuracy, originality, and final form of this appendix rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

COMPANION FILES

Editable forms in Word, spreadsheet, and Markdown, all carrying the same field names, together with one assignment documented from beginning to end, are in the Applied Business Analytics companion repository.

D.1 What the Record Is For

Chapter 1 introduced four habits for AI-assisted work — specify, predict-then-verify, explain, document — and gave the fourth a single sentence: documentation should record the tool used, the prompt, the output, the revisions made, the errors or limitations found, and the verification performed. This appendix is that sentence turned into a form, and it is worth being clear about why the form is shaped the way it is.

The obvious purpose — disclosure — is the least important one. A reader who learns only that an assistant was used has learned almost nothing, because the interesting question is never whether AI contributed. It is whether anyone checked. A record that says "used ChatGPT for the code" discloses a fact and establishes nothing; a record that says which cell, what was predicted before it ran, what the row count actually did, and what changed as a result establishes that the work was audited by a person who can be asked about it.

In other words, this appendix documents verification, and mentions AI because AI is what made the verification necessary. That is why the fields carrying the most weight are the prediction, the error found, and the check performed, and why the field naming the tool is one of eleven rather than the point of the exercise. It is also why this record is not a citation: citing a generative tool is a separate obligation with its own conventions, which change as the tools do (McAdoo, 2023). Underneath all of it is a division of labor the guide has been building since Chapter 1. AI changes which parts of analytical work are cheap and which are scarce, and the scarce part is judgment about what an output means and whether it can be trusted (Davenport et al., 2020). Fields D6 through D11 are where that judgment leaves a trace.

The second purpose is narrower. Analytical work is revisited: a number is questioned in a meeting, a model is reused for a decision it was never graded on, a chart is quoted with a stronger verb than it can carry. The question then is always some form of "how do we know this?" The record is the answer, and it has to be legible to someone who was not in the room — which, six weeks later, includes you.

> **CONCEPT**
>
> **The Record Is Evidence, Not Disclosure**
>
> A completed record does not establish that the work is correct. It establishes that the work was checked, by a named person, against something specific, before it was submitted. Those are different claims, and only the second one a form can support.
>
> The practical test is Chapter 1's: could you answer, without rerunning anything, why the submitted file differs from what the assistant returned? If yes, the record is doing its job. If no, the record is a receipt.
>
> Source: Course concept developed for this guide, informed by the documentation and human–AI configuration practices in National Institute of Standards and Technology (2024).

D.2 The Two Levels

The record comes in two levels because AI-assisted work happens at two scales. A student debugging a `KeyError` needs a line of record beside the cell; a student submitting a churn model with a threshold derived from stated costs needs a document. One form cannot serve both without being too heavy for the first case or too thin for the second.

Both levels use the same field names, and the compact record is a strict subset of the full one. A notebook full of compact records therefore supplies the evidence needed to assemble the exchange log of Section D.5.4 without reconstructing the exchanges from memory, which is the whole reason the field names are held constant across the two.

Table D.2

The two levels compared

	Level 1: compact notebook record	Level 2: full submission record
Where it lives	In a text cell beside the code, chart, or paragraph it produced	A separate document, or an appendix to the deliverable
Unit	One exchange	One assignment or project, covering every meaningful exchange
Fields	Exchange ID plus Chapter 1's four questions; Prediction recorded first from Chapter 4 onward	All eleven fields of Table D.4, plus tools, privacy, verification, dispositions, and the signature
Written	While the work is happening	Assembled at submission from the compact records
Introduced in	Chapter 1, Table 1.6	Chapter 1, Section 1.7
Required by	Every lab from Chapter 4 onward	Every AI-assisted homework submission

D.2.1 Level 1: The Compact Notebook Record

Level 1 is Table 1.6 of Chapter 1: four questions, answered in a text cell next to the work. It exists because the fields that matter most decay fastest. Nobody remembers, two weeks later, what they predicted the row count would be, and a prediction reconstructed after the fact is not a prediction.

From Chapter 4 onward the labs add a fifth line, because those chapters require the prediction to be written before a delegated step runs. Section D.4 gives the form.

D.2.2 Level 2: The Full Submission Record

Level 2 is the document submitted with graded work. It consolidates the meaningful exchanges rather than every conversational turn, adds the tools and the data declaration, collects the verification evidence in one table, requires an account of what was rejected, and ends with a signed statement. Section D.5 gives the form, section by section.

"Meaningful" is doing real work in that sentence. Three rounds of the same debugging exchange are one entry, provided the final prompt, the correction, and the check are preserved. A threshold you accepted from an assistant is one entry on its own, however short the exchange was, because a threshold is a cost assumption and Chapter 9 requires it to be re-derivable.

D.2.3 The Proportional Documentation Rule

How much record a given exchange needs follows from the consequence of getting it wrong, not from how long the exchange was. A twenty-minute conversation about how to phrase an axis label may need one line; a three-line exchange that set a decision threshold needs the full treatment.

Table D.3

The proportional documentation rule

The exchange…	Record	Because
Suggested a syntax correction that changed no number	Compact record, one line, with the check that confirmed nothing else moved	The failure mode is a silent side effect, and one rerun catches it
Drafted a step that changed a count, a total, or a grain	Compact record with the predicted and actual figures, carried into the exchange log	Chapter 4 makes the before-and-after counts non-negotiable
Proposed a definition, a threshold, an eligibility rule, or a window	Full entry, with the choice independently re-derived or checked against the governing source, business rule, or arithmetic	These are the choices the tool is least entitled to make and the meeting is most likely to question

The exchange…	Record	Because
Produced a model, a forecast, an experiment analysis, or a scored list	Full entry, plus its own rows in the verification table	The audit is the deliverable in Chapters 8 through 11
Wrote or compressed a sentence that carries a finding	Full entry, with the clause-by-clause comparison Chapter 13 requires	Compression removes intervals and conditions first
Was one of several near-identical debugging turns	One grouped entry, preserving the final prompt, the correction, and the check	Grouping is permitted for low consequence and never for high

CONCEPT

Documentation Depth Follows Consequence

Ask what would go wrong if this exchange were wrong and nobody noticed. If the answer is a broken cell, one line is enough. If the answer is a budget, a customer treated differently, or a sentence a director will repeat, the answer is the full entry.

Grouping low-consequence turns is allowed and sensible. Grouping so broadly that the verification disappears is not, and it is the most common way a long record becomes an empty one.

Source: Course concept developed for this guide.

D.3 The Canonical Field Set

Table D.4 defines every field once. The Word form, the spreadsheet, and the Markdown version in the companion repository are generated from this table, so a field cannot be renamed in one format and not the others, and a record completed in one format can be read by someone who knows only another.

The right-hand column shows where each field comes from. Four of the eleven are Chapter 1's Table 1.6 questions, reproduced in the same words. Three more are the fields the note under Table 1.6 promises this appendix would add. The remaining four — the identifier, the stage, the prediction, and the change to the final work — are what the labs from Chapter 4 onward actually require, and they are the reason a record assembled from compact notes is auditable rather than merely complete.

Table D.4

The canonical field set, and where each field comes from

Field	What goes in it	Chapter 1 source
Exchange ID	A short stable label — E1, E2, E3 — written beside the cell, view, or paragraph the exchange produced, so the record and the work point at each other.	—
Stage	Specify, prepare, analyze, visualize, explain, document, or review. One word; it is what makes a long log searchable.	—
Tool and version	The product name and whatever version string the interface displays. Record the access mode too when it is not the obvious one.	Which AI tool did you use, and for which step?
Prompt as sent	The prompt in the words you actually sent, not a tidied reconstruction. Long prompts may be attached and referenced by location.	What prompt did you submit?
Output received	What came back, summarized in a sentence or two, with the location of the full response if you kept it.	— (promised by the Table 1.6 note)
Prediction recorded first	What you wrote down before the output was run or read: the row count, the total, the shape, the direction, the band. Blank is a finding.	—

Field	What goes in it	Chapter 1 source
Decision	Accepted, modified, rejected, or used only to generate questions. One of the four; not a paragraph.	What did you accept, revise, or reject?
Error or limitation found	Technical, analytical, factual, interpretive, or communication. Name the class and the instance.	— (promised by the Table 1.6 note)
Verification performed	The check you ran and the result: the reconciled total, the hand calculation, the rerun, the baseline, the source consulted, the reader tested.	How did you verify the final result?
Change to the final work	What is different in the submitted artifact because of this exchange. If nothing changed, say so and say why you kept it.	—
Analyst judgment, and what it taught you	The decision you made that the tool could not make, and the one thing the exchange changed about how you will prompt or check next time.	— (promised by the Table 1.6 note)

One field deserves a note. **Prediction recorded first** is blank on exactly two kinds of exchange: those that produced no output to predict, and those where the analyst forgot. The form does not distinguish them, so write "none required" in the first case. A log with many blank predictions is a finding about the analyst's process, and it is better to see it than to fill it in afterward.

D.4 Level 1: The Compact Notebook Record

Table D.5 is the form. Copy it into a text cell directly beneath the cell, chart, or paragraph the exchange produced, and give it an exchange identifier that the full record can refer to later.

The labels are the canonical ones from Table D.4, not a second vocabulary, which is what lets a notebook full of these supply the exchange log directly rather than from memory. Chapter 1 poses them as four questions, and those questions are the gloss rather than the field names: *Which AI tool did you use, and for which step?* is **Tool and version**; *What prompt did you submit?* is **Prompt as sent**; *What did you accept, revise, or reject?* is **Decision**, whose permitted values are the four of Table D.4 — modified is the word for revised; and *How did you verify the final result?* is **Verification performed**. **Prediction recorded first** is the fifth line, required from Chapter 4 onward wherever a delegated step changes a count, a total, or a grain.

Table D.5

Level 1, the compact notebook record

Field	Your response
Exchange ID	[E1]
Tool and version	
Prompt as sent	
Prediction recorded first	
Decision	
Verification performed	

A completed one looks like this, from a Chapter 4 cleaning step on the practice file. It is six lines and it took under a minute to write.

Example D.1. A completed compact record, in a notebook text cell

```
Exchange ID: E3
Tool and version: general-purpose assistant, version not
  displayed; drafted the channel-standardizing cell below.
Prompt as sent: "Standardize the channel labels in
  transactions_raw to Online, App, Store. Print the value counts
```

```
  before and after and state the row count you expect. Change
  nothing else."
Prediction recorded first: 5 labels collapse to 3; row count
  unchanged at 14; revenue total unchanged at $767.40.
Decision: Modified.
Verification performed: the response mapped "web" to Online,
  which is right, and silently dropped two rows whose channel was
  missing, which I did not ask for; I removed that operation.
  After the correction: labels 5 -> 3, row count 14 on both
  sides, revenue $767.40 on both sides.
```

Note what makes it useful, and note that all its field labels come from the full record. The prompt is quoted rather than described. The prediction names three quantities and the verification names the same three. The decision is one of the four permitted words and nothing else; the defect it responds to — a silent row drop, which Section 4.10 names as the first of the three cleaning failure modes — sits in the field built to carry it. Someone reading the notebook can reproduce the check in one cell.

D.5 Level 2: The Full Submission Record

The full record has eight parts, completed in the order given. The first three establish who did the work, with what, on what data. The fourth is the log. The fifth is the evidence. The sixth is what did not survive intact. The seventh is the signature, and the eighth applies only to group work.

The tables that follow define each part, field by field, rather than presenting a form you write into on the page. The fillable versions — in Word, spreadsheet, and Markdown, carrying these field names and nothing else — are in the companion repository, and Section D.8 says which to use.

D.5.1 Assignment and Analyst Information

This section is short and is completed first, because a record that cannot be attached to a specific version of a specific artifact is not evidence about anything.

Table D.6

Assignment and analyst information

Field	Response
Student name	
Group name and members, if applicable	
Course and section	INTG1-GC 2300, Section [SECTION]
Assignment or project	
Submission date	
Notebook, workbook, dashboard, or report filename	
Version of the submitted artifact	

D.5.2 AI Tools Used

One block per tool. If you used three assistants for three different purposes, that is three blocks, and the purposes matter more than the names: a tool used for adversarial review carries a different risk profile from the same tool used to draft prose that will carry a finding.

Table D.7

AI tools used, one block per tool

Field	What goes in it
Product name	Whatever the interface calls itself.
Model or version	The version string displayed, if any. Write “not displayed” rather than guessing.
Access mode	Public web tool, institutional account, embedded in Colab, embedded in another application, or local.
Dates used	First and last date, or the single date.
Purpose categories	One or more of the nine categories in Table D.8.

Table D.8 lists the nine purpose categories the guide uses, with the chapters in which each is used or assessed. Use these words rather than inventing your own, so that a term-long log can be read across assignments.

The categories are not the same thing as the **Stage** field of Table D.4, and the two are easy to conflate. Stage identifies where in the workflow the exchange occurred; purpose category identifies what kind of contribution the assistant made. One tool used across a whole assignment carries several purpose categories in this block, while each exchange in the log carries exactly one stage.

Table D.8

Purpose categories, and where each is used or assessed

Category	What it covers	Chapters
Problem framing	Turning a request into a specification; red-teaming a specification you wrote	1–2
Data preparation	Drafting a cleaning or joining step whose effect on counts and totals you predicted first	4
Code drafting	Writing a cell you specified, in a language you can read	1, 4–11
Debugging	Explaining an error message and proposing the smallest correction	4–11
Method explanation	Explaining what a method does, in terms you can check against the chapter	1, 3–11
Analytical interpretation	Proposing readings of an output you have already verified	5–11
Visualization design	Proposing chart specifications, encodings, layouts, or titles	12–13
Writing or editing	Drafting or compressing prose that carries a finding	13
Adversarial review	Arguing against your own specification, model, chart, or recommendation	2, 7–13

D.5.3 Data and Privacy Declaration

Section 1.13 states the rule the course actually enforces: confidential and personally identifiable data are never to be placed into an external AI tool. The declaration is where you record what you supplied and under what conditions. Chapter 3 adds the reason it belongs on the form rather than in a policy nobody rereads — pasting customer-level data into an external assistant is a custody decision rather than a convenience, and the granularity of what is collected and where it travels is itself a privacy design choice (Martin & Murphy, 2017).

Table D.9

Data and privacy declaration

Question	What a complete answer contains
What information was supplied to the tool?	Name the tables, columns, and row counts, or the schema alone. “The data” is not an answer.

Question	What a complete answer contains
What kind of information was it?	Synthetic, public, course-provided, de-identified, or confidential. The StyleCraft files are synthetic and are safe to paste.
Were any personal identifiers, credentials, proprietary records, or restricted data supplied?	Yes or no. If yes, name the approved environment that permitted it, or report the incident to your instructor.
What safeguards were used?	Schema only, safe sample, synthetic substitute, de-identification, aggregation, or an approved enterprise tool.

The section then closes with exactly one of the three statements in Table D.10, signed. They are mutually exclusive on purpose. A single confirmation qualified by *except as described above* would let an incident be described and then signed away in the same sentence, which is the opposite of what a declaration is for.

Table D.10

The three declarations. Sign exactly one

Statement	Sign it when
1. No restricted information was supplied.	Nothing confidential, personally identifiable, proprietary, or otherwise restricted was placed into an AI tool at any point in this work.
2. Restricted information was used only within the approved environment named above.	The environment, the permission under which it was used, and the information supplied are all recorded in the table above.
3. A possible incident occurred and has been reported.	Describe it above, report it to your instructor, and do not sign statement 1 or 2. Reporting an incident is not a penalty; concealing one is a different matter.

The StyleCraft files are synthetic, so for coursework the honest answer is almost always the first. The habit is what transfers — and so is the third, because an organization learns about an exposure from the person who caused it or from someone else, and the second route is always worse.

D.5.4 The Exchange Log

One entry per meaningful exchange, using the eleven fields of Table D.4. The shape follows the format rather than the other way round. In a spreadsheet, one exchange is one row and the eleven fields are columns, which is what makes a long log sortable. In Word, one exchange is a vertical two-column block, field beside entry; an eleven-column table on a portrait page is unreadable and worse to complete. In Markdown either shape works, and short logs read better as blocks. Section D.8 covers the formats, and the field names do not change between them.

Three rules govern the log. Preserve the prompt as sent, because a polished reconstruction hides how underspecified the original request was — which is the whole point of the deliberately loose prompts Chapters 4 through 13 ask you to submit verbatim. Match the identifier in the log to the identifier in the notebook, workbook, or memo; a log that cannot be tied to the work it describes is a parallel document. And keep the decision field to one of the four permitted words, putting the reasoning in the fields built for it.

Several chapters run a two-prompt pattern: a first prompt that produces mechanics and ends by forbidding narration, then the analyst's audit, then a second prompt that narrates within the audited results. File both, as adjacent rows sharing a stem — E7a and E7b — with the audit recorded between them in the verification table. Filing them as one row loses the audit, which is the part the pattern exists to protect.

D.5.5 Verification Evidence

This is the table a reader turns to first, and in most submissions it is where the grade is. It carries the checks that mattered, not every `assert` in the notebook — six well-chosen rows beat forty mechanical ones.

Table D.11

The verification evidence table

Field	What goes in it
Claim or output checked	The specific number, table, chart, or sentence under test — not the assignment.
Prediction, recorded first	What you expected, written before the check ran. A prediction written afterward verifies nothing.
Verification method	Reconciliation, hand calculation, rerun, baseline comparison, holdout, source check, reader test.
Expected value or condition	The figure or the condition that would count as a pass, stated in advance.
Actual result	What the check returned, in the same units.
Verdict	Pass, fail, or unresolved. Unresolved is a permitted verdict; a silent one is not.
Action taken	What you did about it. A failed check with no action is an unrepaired defect, not a completed record.
Evidence location	Where the evidence lives: cell, sheet, page, commit, file name.

CONCEPT

The Prediction Must Be Older Than the Result

The second column of Table D.11 is the one that makes the table worth keeping. A check whose expected value was written after the actual value was seen cannot fail, and a check that cannot fail is not a check.

This is Section 1.7's second habit, written down. Predict the number, the range, the shape, or the direction; run the step; compare; and when they disagree, record the disagreement rather than the resolution alone. Chapter 12 makes the same point from the other side: a fault you predicted and did not find is as informative as one you missed.

Source: Course concept developed for this guide, informed by the verification discipline of Chapman et al. (2000) and Provost and Fawcett (2013).

The evidence the guide's labs actually produce takes many forms, and the table is built to hold all of them. Table D.12 names the kinds, with the chapter that introduces each, so that "verification performed" does not collapse into "I looked at it."

Table D.12

Kinds of verification evidence this table is built to hold

Kind	What the row records	Introduced in
Row-count and total reconciliation	Counts and totals before and after a step, and the difference explained	Chapter 4
Hand calculation	A figure computed with a calculator on a miniature, beside the computed one	Chapters 3 and 5
Known-truth comparison	The answer the dataset was built to contain, recovered or not	Chapters 4, 6, and 11
Baseline comparison	The opponent's score on the same data, and the margin	Chapters 8 and 10
Holdout or backtest	A sealed evaluation opened once, after a declared selection rule	Chapters 8, 9, and 10
Confusion-matrix arithmetic	The four cells re-added by hand, summing to the evaluated population	Chapter 9

Kind	What the row records	Introduced in
Threshold re-derivation	The cut recomputed from the stated costs, with its sensitivity	Chapter 9
Experiment assignment check	The flow from randomization to outcome, planned against realized	Chapter 11
Chart audit	Grain, axis floor, encoding, filter state, and the reading each changed	Chapter 12
Reader test	A three-second or ninety-second test, with the reader's words recorded verbatim	Chapter 13
Clause-by-clause comparison	A compressed paragraph diffed against the original, each removal classified	Chapter 13

D.5.6 Rejected, Corrected, or Bounded Output

Most AI-assisted submissions in this course will carry at least one entry here, and several chapter exercises require one in their own words: at least one delegated step you corrected and why, at least one summary whose denominator you had to fix, or at least one case where the assistant's prediction and yours disagreed. Table D.13 is where they go, and its second field is what keeps the section honest about what actually happened.

Table D.13

Rejected, corrected, or bounded output

Field	What goes in it
Output, claim, limitation, or suggestion	Quote it. A paraphrase usually removes the thing that made it wrong, or the ambiguity that made it worth recording.
Disposition	Rejected, corrected, limited, or declined. One of the four.
Reason for disposition	The specific defect, ambiguity, or boundary, named.
Evidence used	What you checked it against.
What appears in the final work	The replacement, the correction, the stated limit, or the fact that nothing from this exchange was used.

The four dispositions are why this section is not called *errors*. **Rejected** means the output never entered the work; **corrected**, that it entered after a specific repair; **limited**, that it entered with a stated boundary on what it may be used for; **declined**, that you considered a suggestion and did not take it. So the purpose is not to manufacture an error, and a fabricated one is worse than none. If nothing material went wrong, record the limitation you worked around or the ambiguity the assistant resolved without saying it was choosing — Chapter 3's exercise puts that question well: which definition did it choose, and did it state that it was choosing?

Where an exercise does not require an entry and nothing was rejected, corrected, limited, or declined, the section is not left blank. Write *none identified* and name the checks that would have detected a material defect. That converts an empty section into a claim someone can evaluate: "I found nothing" is a finding when the checks behind it are listed and an absence otherwise. Chapters 11, 12, and 13 make the same point from the other direction — an exchange in which the assistant performed badly and you caught everything is a better submission than one in which it performed well — and none of them asks you to go looking for something to reject.

D.5.7 The Analyst-of-Record Statement

The record ends with the statement below, adapted to your own work. It covers five things: what the tool contributed, what you contributed, how the final work was verified, what limitations remain, and that you can explain what you submitted. The bracketed clause is the one that changes every time; the rest is a fixed form so that a reader can find the limitations in the same place in every submission. Then your name and the date.

Statement D.1. The analyst-of-record statement

```
I remain responsible for the submitted analysis. I reviewed the
AI-assisted output identified above, made the decisions recorded in
this record, and verified the final work using the evidence listed.
I can explain the code, the calculations, the visualizations, and
the recommendations in this submission. The remaining limitations
are [LIMITATIONS].

Name: [NAME]                                Date: [DATE]
```

CONCEPT

Sign What You Can Explain

Chapter 1 set the standard in one line: do not send work to a manager that you cannot explain. The statement above is where that becomes a signature, and the clause that carries the weight is the third one.

Before signing, run Section 1.9's four questions on the finished artifact. Does the output answer the question I specified? Does one row still represent what I said it represents? Can I reproduce at least one number by hand? Can I state in one sentence what this result does not show? If any answer is no, the work is not ready and the statement is not yet true.

Source: Course concept developed for this guide.

D.5.8 Group Contribution Record

Group projects submit one consolidated record rather than several disconnected forms, with the contribution table below appended. The point is traceability, not accounting: a reader should be able to take any exchange in the log and find the member who ran it and the member who checked it.

Table D.14

Group contribution record, one row per member

Field	What goes in it
Member	Name.
AI-assisted task performed	Which exchanges, by ID.
Verification performed	Which checks in the verification table are this member's.
Final artifact contribution	The cells, views, sections, or paragraphs that carry this member's work.
Peer confirmation	The initials of a second member who has read the row and agrees with it.

Two group-specific rules follow from the rest of this appendix. Exchange identifiers are unique across the group, so E7 means one thing in the consolidated log. And the member who ran an exchange should not be the member who signs off on its verification wherever the work allows otherwise — Chapter 1's second-reviewer practice, applied to a team that already has a second reader available.

D.6 A Weak Record and a Strong One

The difference between a record that helps and a record that occupies space is almost always specificity. Table D.15 sets one against the other, on the same exchange: a Chapter 5 summary table drafted by an assistant on the fourteen-line miniature Lab 5.1 certifies at eight orders and $660.00 of revenue.

Table D.15

The same exchange, recorded weakly and recorded well

Field	Weak	Strong
Tool and version	ChatGPT	General-purpose assistant; version not displayed; public web tool. Drafted the channel-summary table in cell 12
Prompt as sent	Asked it to summarize the data	"Summarize this sales data and tell me what it means" — the deliberately vague prompt of Exercise 5.6, sent verbatim
Prediction recorded first	—	Revenue subtotals sum to $660.00; eight orders; average order value near $82.50, not the average of the group means
Decision	Used it	Modified
Error or limitation found	Some small issues	Analytical: reported an overall average order value of $105.00 by averaging the two group means, and stated no denominator for the channel shares
Verification performed	Checked it	Recomputed the pooled figure by hand: $660.00 ÷ 8 = $82.50. The weighted recombination returns $82.50; the averaged average returns $105.00
Change to the final work	—	Replaced the headline with the pooled $82.50, added the denominator to every share, and struck the causal verbs from its narration
Analyst judgment, and what it taught you	—	Pooled, because the question is about the chain rather than the average branch. Next time the prompt states the denominator before the assistant picks one

The weak record is not shorter by accident. Every one of its entries is a place where a specific answer existed and a general one was written instead: "some small issues" had a name, "checked it" had an arithmetic, and "used it" concealed the fact that the headline number was wrong and was fixed. Notice also what the strong record costs — about a hundred and twenty words on a two-minute exchange, every figure of which was already on the screen when it was written. That is the practical argument for writing the compact record during the work. At that moment it is nearly free; an hour later it is archaeology.

D.7 Retention and Evidence Links

The record points at evidence; it does not contain all of it. A prompt that runs to two pages is attached and referenced by location, and a reconciliation is cited by the cell that printed it rather than transcribed. Table D.16 gives the link types the course accepts and what each is good for.

Table D.16

Evidence links, and what each is good for

Link type	Good for, and what to watch
Notebook cell	Printed counts, totals, assertions, and charts. Cell numbers move, so cite the exchange ID you wrote beside the cell rather than its position
Saved conversation export	A long prompt, or a response you needed in full. Export where the tool and the course permit it, and never export restricted data
Workbook or Tableau worksheet	Reconciliation tables and view-level provenance. Name the sheet; "the workbook" is not a location
Version-control commit	Showing what changed and when. Cite the short hash and the file, not the branch
Report or memo page	A claim, a caveat, or a reversal condition. Page numbers move between drafts, so cite the version too

Link type	Good for, and what to watch
Screenshot	An interface state that leaves no other trace, such as a filter. A screenshot is evidence of an appearance, not of a number

One rule overrides the table. Evidence must travel with the submission or remain reachable through the course-approved environment. A link to a conversation on a personal account, a file in a drive nobody else can open, or a page that may not exist next term is not evidence; it is a promise. Where a tool does not permit export, summarize the response in the log and say that no export was available.

Retention has two halves that pull against each other. Keep the record and its evidence available for as long as the work can be questioned — through the course's grading and appeal period here, and for whatever period the organization requires in professional work. Then delete what you no longer need, particularly exported conversations and any sample of data that was a convenience rather than a requirement. A record is evidence for a period; an unnecessary export of restricted data is a liability indefinitely.

D.8 Completing the Record in Word, a Spreadsheet, or Markdown

The same record can be completed in any of three formats, and the choice is a question of what the submission is rather than of preference. What does not change is the field set: the same eleven exchange fields, the same eight verification fields, the same declaration, the same statement. The companion repository carries all three, generated from one source.

The three files share one stem, `Appendix_D_AI_Use_Record_FORM`, and differ only in extension. Use the `.docx` when the submission is a narrative assignment, a memo appendix, or a project deliverable. Use the `.md` when the record lives inside a Colab notebook or a repository: it survives export to `.ipynb`, renders in the notebook itself, and diffs cleanly under version control. Use the `.xlsx` — or the CSV files corresponding to its worksheets — for an assignment with more than about a dozen exchanges, where one row per exchange sorts, filters, and counts in a way a Word table does not. Group work with simultaneous editors can open the Word and spreadsheet files in Google Drive without changing a field name.

Whichever format you use, the submitted record should be a file rather than a screenshot of a file, for the same reason Section A.15 asks for notebooks rather than pictures of notebooks. A record that cannot be searched cannot be audited.

D.9 What Each Chapter Asks You to File

Every chapter of the guide requires this record, and each one narrows it to the exchange its own work makes most consequential. Table D.17 states what each chapter asks for and which level satisfies it. Where a chapter asks for something the general form does not name — a count of `UNSPECIFIED` rows, a graded prediction including false alarms, a clause-by-clause diff — it belongs in the verification table, as a row whose method is that chapter's audit.

Table D.17

What each chapter asks you to file

Ch.	What the chapter asks to be filed	Level
1	Section 1.7 and the Review Before You Rely box; Exercise 1.6 asks for the compact record in the notebook and the full record for submission	1 and 2
2	Section 2.8's adversarial review of your own specification; Exercise 2.7 asks for the full record	2
3	Section 3.9's drafted dictionary entries and the audit of them; Exercise 3.7 asks for the full record	2
4	Every delegated cleaning step, with the assistant's predicted effect on counts beside yours; Exercise 4.5 requires at least one recorded disagreement	1 in the notebook, 2 for submission

Ch.	What the chapter asks to be filed	Level
5	Every drafted summary, with the denominator, the center, and the reconciliation audited; Exercise 5.5 requires at least one corrected summary	1 in the notebook, 2 for submission
6	The feature build and the cluster audit; Exercise 6.6 requires the full exchange and which audit step changed your assessment most	2
7	Both prompts of the two-prompt pattern, with the verbs graded against Section 7.2's registry; Exercise 7.6 requires the full exchange	2
8	Both exchanges, plus the five-point audit written out in order; Exercise 8.6 requires every performance claim marked earned or not earned	2
9	Both exchanges, plus the classification supplement; Exercise 9.6 requires the loose-prompt response and the re-prompted one, compared	2
10	Both exchanges, plus the leaderboard band you predicted before reading the response; Exercise 10.6 requires the comparison of the two prompts	2
11	Four moves — registration, analysis, adversarial review, drafting — each filed; Exercise 11.6 requires the UNSPECIFIED count and the marked-up recommendation	2
12	Both exchanges of the five-step routine, plus the predicted faults, the graded prediction including false alarms, and a note per repair	2
13	The exchanges — title generation and prose compression are separate tasks — with the candidates' verification lines and the clause-by-clause diff; the record travels inside the memo's appendix	2

Two patterns run through the table. The first is that the unit of filing grows with the work. Chapters 7 through 10 run the two-prompt pattern — mechanics, the analyst's audit, then narration — and ask for both prompts rather than the final one, because the audit between them is what is being graded. Chapter 11 expands that into four moves: registration, analysis, adversarial review, and drafting, each filed. Chapters 12 and 13 run a five-step visual and communication routine in which three of the five steps are analyst work rather than exchanges, and the prediction is graded including its false alarms. The second pattern is that from Chapter 8 onward the record stops being an attachment and becomes part of the deliverable: the audit memo is the lab's final deliverable in Chapters 8 through 11, and in Chapter 13 the record is filed inside the recommendation memo's appendix. That progression is this appendix's real argument. In Chapter 1 the record is a form you complete because the course asks; by Chapter 13 it is what lets a reader see not only what you concluded, but how you came to be entitled to conclude it.

References

Every work below is cited above. Chapter 1 remains the guide's own statement of the four habits, the compact record, and the privacy rule.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide*. SPSS Inc.

Davenport, T., Guha, A., Grewal, D., & Bressgott, T. (2020). How artificial intelligence will change the future of marketing. *Journal of the Academy of Marketing Science, 48*(1), 24–42. https://doi.org/10.1007/s11747-019-00696-0

Martin, K. D., & Murphy, P. E. (2017). The role of data privacy in marketing. *Journal of the Academy of Marketing Science, 45*(2), 135–155. https://doi.org/10.1007/s11747-016-0495-4

McAdoo, T. (2023, April 7). *How to cite ChatGPT*. APA Style. https://apastyle.apa.org/blog/how-to-cite-chatgpt

National Institute of Standards and Technology. (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile* (NIST AI 600-1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.600-1

Provost, F., & Fawcett, T. (2013). *Data science for business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

Appendix E

Data Visualization Checklist

Dr. Jose Mendoza, Academic Director and Clinical Associate Professor
Version 1.0 · August 2026
Except where otherwise noted, this appendix is licensed under CC BY 4.0.

Appendix Information

PURPOSE

This appendix is the last check a chart, an analytical pre-read, an executive dashboard, or a recommendation memo passes before it leaves the analyst's workspace. It consolidates the visual-mechanics obligations of Chapter 12 with the audience, hierarchy, uncertainty, usability, and ethical obligations of Chapter 13 into fourteen checkpoints that can be run rather than remembered, sorts every check into the ones that stop an artifact and the ones that are judgment calls, and ends in a signed record of what failed, what was repaired, and what the analyst decided to accept.

VERSION AND DATE

Version 1.0 · August 2026 · Language: English (United States)

SUGGESTED CITATION

Mendoza, J. (2026). Data visualization checklist. In *Applied business analytics for marketing decision-making: Business analytics and data visualization* (Appendix E, Version 1.0) [Open educational resource]. CC BY 4.0.

LICENSE AND RIGHTS

Except where otherwise noted, this appendix is licensed under a Creative Commons Attribution 4.0 International License.
Copyright © 2026 by Jose Mendoza.

Tableau, Tableau Desktop, and Tableau Prep are trademarks of Salesforce, Inc. Google Colab and Google Drive are products of Google LLC. ChatGPT is a product of OpenAI. Claude is a product of Anthropic. Gemini is a product of Google LLC. GitHub Copilot is a product of GitHub, Inc. "Python" and the Python logos are trademarks or registered trademarks of the Python Software Foundation. Matplotlib and pandas are NumFOCUS fiscally sponsored projects. Product names are used for identification only and do not imply endorsement. StyleCraft Collective is a fictional company created for instruction.

GENERATIVE AI USE

Generative artificial intelligence and other AI-assisted tools were used in the research, writing, revision, and production of this appendix, including outlining, preliminary drafts, revision of prose, and document formatting. These tools were used under the author's direction and are not credited as authors, researchers, or sources. The author determined the appendix's scope, boundaries, and content, and reviewed and approved all AI-assisted material: every check was traced to the chapter section that establishes the obligation it enforces, and every figure quoted from the StyleCraft case was reconciled against the chapter that derives it rather than restated from memory. Responsibility for the accuracy, originality, and final form of this appendix rests entirely with the author. A fuller statement appears in the front matter of the complete guide.

COMPANION FILES

A printable one-page checklist, an editable Word and Google Docs version, a spreadsheet version carrying the Stop, Caution, and Pass status column, a Tableau-project sign-off form, and an accessible plain-text checklist are in the Applied Business Analytics companion repository.

E.1 What This Checklist Is For

Chapter 12 established that a chart can produce a false impression without containing a false statement, and that the professional standard is therefore not *did I intend to mislead* — a question about the analyst's interior state that nobody else can check — but *did I check*, which leaves evidence. Chapter 13 added a second question on top of the first: not only whether the artifact shows what the data contains, but whether the reader took away what the artifact

actually supports. This appendix exists because both standards fail in exactly the same circumstance, which is a deadline.

An obligation that depends on remembering to care will be honored on quiet weeks and abandoned on the weeks that matter, and the weeks that matter are the ones in which charts reach decision-makers. The merchandising analyst's deck in Section 12.1 was not built in bad faith; it was built in ninety minutes by a capable person who accepted four software defaults and had no framework for knowing which defaults were dangerous. Making the framework a checklist — read the axis floor, name the grain, name the comparison, run the grayscale check, write the footer — removes it from the domain of virtue and puts it in the domain of process, which is where obligations survive.

CONCEPT

Every Box Checked Is Not a Certification

This checklist can establish that an artifact was inspected against fourteen classes of known failure. It cannot establish that the artifact is correct, that the analysis beneath it is sound, or that the recommendation it supports is the right one. Those are analytical judgments, and a checklist that claimed to make them would be doing the thing this guide has warned against since Chapter 1, which is treating a completed procedure as evidence that thinking occurred.

What a completed checklist is good for is narrower and genuinely valuable. It makes the analyst's inspection auditable, it makes the accepted defects visible as decisions rather than as oversights, and it converts a recurring argument about whether a bar chart may start at 4,000 into a settled rule that was decided once, in a document, by people who were not under deadline. Treat a run of this checklist as a record of what was checked, and treat the sign-off record of Section E.7 as the artifact that record produces.

Source: Course concept developed for this guide, informed by Sections 12.16 and 13.15.

The division of labor is worth stating once, because several checks ask the analyst to confirm something rather than to compute it, and the difference matters. Table E.2 records what the checklist decides and what it does not.

Table E.2

What the checklist settles and what remains the analyst's

The checklist settles	The analyst decides	Established in
Whether a length-encoded mark begins at zero	Whether the finding survives its honest rendering, and whether there is a finding at all	Sections 12.6, 12.7
Whether the grain of one mark was stated and confirmed	What grain the question requires	Sections 3.3, 12.10
Whether a reference is on the chart and labeled	Which reference the reader needs, and whether it was declared before the results were inspected	Sections 2.7, 12.9
Whether an assertion title is verifiable from the marks beneath it	Which of the defensible sentences to write, and therefore what the reader takes away	Sections 13.6, 13.15
Whether the removals were listed	Whether each removal is a simplification or the deletion of a qualification	Section 13.15
Whether uncertainty is present and in decision units	How much uncertainty the decision can carry, and what would reverse the recommendation	Sections 2.7, 13.9
Whether a reader's takeaway was measured	What to do when the reader's takeaway is not the intended one	Sections 13.2, 13.10

The checklist settles	The analyst decides	Established in
Whether every AI-touched sentence was verified against a view	Whether to keep the sentence	Sections 12.12, 13.11

Read the second column as the reason the first column is short. Almost every entry in it is a decision Chapters 12 and 13 spend pages establishing, and none of them can be made by inspection. The checklist's job is to make sure the decision was made rather than inherited.

E.2 Two Check Levels, Three Status Labels, and One Sign-Off Record

Two distinct things get labeled in this appendix and confusing them is the most common way a checklist becomes useless. A **level** is a property of the check and never changes: it says whether the obligation is a hard error or a judgment call, and there are two. A **status** is the outcome the analyst records on a particular artifact: it says what happened when the check was run, and there are three. A single check therefore has one level and, on each artifact, one status. Running the two levels across an artifact produces the third object in this section's title, the sign-off record of Section E.7, which is a document rather than a level.

E.2.1 The Two Check Levels

Level 1 — stop-ship. A Level 1 check identifies either a defect that makes the artifact incorrect, materially misleading, or unusable, or the absence of a mandatory control that makes the audit record invalid. The first kind is recognized by its effect on a reader: she would believe something about the quantities that the data does not support, and would revise that belief on seeing the evidence. The second kind is recognized by what it does to the audit: a decision never written down, a hierarchy never stated, a fault never predicted before the chart was rendered — none of these misleads anyone by itself, and each one makes it impossible to establish afterward that the artifact was checked rather than merely approved. Both kinds are resolved before the artifact circulates, or the artifact does not circulate, and neither is weighed against a deadline.

Level 2 — communication and judgment. A Level 2 check evaluates whether the intended reader can find, understand, and use the evidence. Its failures are real and are graded against a stated reader rather than against a rule, which means a Level 2 check can be failed defensibly: an analyst who decides that a legend is acceptable here, that this window is the right one, or that this view stays on the canvas has made a decision, and recording the decision with its reason is what makes it a decision rather than an omission.

The sign-off record is what running the two levels produces, and Section E.7 specifies it. It carries what was run, what passed, what failed, what was repaired, what was accepted, and who signed. A run of the two levels that leaves no record has inspected an artifact without leaving evidence that it was inspected, which is the state this appendix exists to replace. The record is not a third level, because nothing in it is a check; it is the output the checks are run to produce.

The two levels do not correspond to importance. A Level 2 hierarchy failure on an executive dashboard — the reader cannot tell what to look at first — will more often cost a decision than a Level 1 units label missing from an axis. They correspond to *how the failure is resolved*: a Level 1 failure has a correct answer, and a Level 2 failure has a defensible one.

E.2.2 The Status Labels

Three statuses are recorded, and all three are written as words. Color may accompany them and may never replace them, both because status communicated by color alone fails for a substantial share of readers and because a printed or grayscale copy of a checklist is exactly the copy somebody will consult under deadline (W3C, 2024; Wong, 2011).

Table E.3

The three status labels and what each one obliges

Status	What it means	What it obliges
Pass	The check was run and the artifact satisfies it.	Nothing further. Record it as run; a check recorded as passed without being run is the failure this appendix cannot detect.
Caution	The check was run, the artifact does not satisfy it, and the analyst has decided to accept the defect. Available on Level 2 checks only.	One sentence naming the defect and the reason for accepting it, in the sign-off record. An accepted defect is legitimate; an unrecorded one is not.
Stop	The check was run and the artifact fails it. Mandatory on any failed Level 1 check.	Repair before circulation, a repair note per Section E.6, and a re-run of the checks the repair touched.

Two outcomes sit outside the three statuses and are recorded rather than scored. A checkpoint the routing of Section E.3 does not send to this artifact is marked **not applicable**, with its reason in three words; “not a dashboard” is a reason and a blank cell is not. And a Level 2 check that failed and was then repaired is recorded as **Pass**, with the change listed under communication revisions in the sign-off record, so that the record distinguishes a check that never failed from one that failed and was fixed.

One asymmetry in that table is deliberate and worth naming, because students reach for the wrong label under pressure. **Caution is not available on a Level 1 check.** There is no defensible version of a truncated magnitude bar, an undisclosed filter that changes the conclusion, a per-order claim computed from per-line rows, or a title asserting a number no view contains. If a Level 1 check fails and the artifact ships anyway, the honest record is a Stop that was overridden, with the name of the person who overrode it — which is a different document from a checklist and a conversation worth having before it becomes necessary.

CONCEPT

The Default Is Not a Defense

Four of Chapter 12’s five misleading mechanics are available as software defaults or single clicks, and the fifth is what a date filter does when nobody chooses a date. That fact is usually offered in mitigation and is the opposite of one: a defense of the form “the tool chose the axis” concedes precisely the thing that matters, which is that nobody looked.

Several checks in this appendix therefore ask you to confirm that a property was *chosen* rather than that it is *correct* — check 5.4 on axis range and check 9.3 on the default filter state at Level 1, check 3.6 on sort order and check 8.10 on dashboard sizing at Level 2. An inherited default that happens to be right still fails those checks, because the artifact will change and the default will follow it. Axis ranges recompute silently on every filter, measure swap, and table calculation. An axis is not a thing an analyst sets once; it is a thing an analyst reads every time the view changes, and the reading takes two seconds.

Source: Course concept developed for this guide, informed by Sections 12.6, 12.7, and 12.16.

E.3 What Runs on What

Running one hundred and eighty-eight individual checks on every chart would guarantee that none of them is run on anything. What the appendix asks for instead is a rule about *when* each of its two instruments is used, and the rule turns on whether a complete audit of this artifact already exists.

The first release of an artifact, and any graded submission, runs the applicable master checkpoints in full. That run is the artifact’s baseline audit, it is what the sign-off record’s counts are taken from, and there is no shorter route to it: a defect nobody has looked for is not caught by an instrument that does not ask about it. **Every**

run after that is a short form. Once a baseline audit exists, the short form of Section E.5 is the working instrument — for routing attention on a revision, for the routine re-run after a repair, and for the check before recirculating something already audited — and a short-form question that fails, or whose answer is not obvious, sends the analyst back into the master checkpoint it cites.

The division is not a compromise between rigor and time. A short form consolidates several master checks into one question, which is exactly what makes it fast and exactly why it cannot substitute for a first look: a question about axes will catch a truncated bar on a revision, and it will not by itself surface a filter that was never disclosed, because nobody asked. Table E.4 therefore names both — the short form each artifact class uses, and the master checkpoints the baseline audit covers. The timings are for the short form; the baseline audit takes longer and takes it once.

Table E.4

Which instrument runs on which artifact

Artifact	Short form	Baseline audit covers	Conditional additions	Short-form time
Python chart inside a working notebook (Chapters 4–11)	E.5.1	E.1 (1.1, 1.3), E.2, E.3, E.4, E.5, E.6, E.7	E.13 if AI-assisted; E.14 if exported or submitted	Under two minutes
Tableau analytical view or pre-read (Chapter 12)	E.5.2	E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.11 (11.5, 11.11), E.14	E.9 if the workbook carries controls; E.13 if AI-assisted	Ten to fifteen minutes
Executive dashboard or decision artifact (Chapter 13)	E.5.3	E.1 to E.8, E.10, E.11, E.12, E.14	E.9 if interactive; E.13 if AI-assisted	Thirty to forty minutes, plus the reader test
Recommendation memo paired with a visual artifact	E.5.4	E.1 (1.1, 1.2, 1.6), E.2 (2.13, 2.15), E.6, E.10, E.11 (11.1, 11.2), E.12 (12.12)	E.13 if AI-assisted	Ten minutes, plus the third-reader test

Four notes on the routing. The first is that the checkpoints are ordered as they are because the order is the cheapest one: E.2 comes before E.3 because there is no point auditing the chart form of a view built at the wrong grain, and E.12 comes near the end because a reader test on an artifact that has not passed E.1 through E.11 measures the wrong thing. Run them in order.

The second is that two checkpoints are conditional rather than universal, including on a dashboard. Checkpoint E.9 runs only where a reader can change the artifact; a static decision report, a printed board pack, or a PDF export has no interactivity to audit, and marking E.9 not applicable there is correct rather than lax. Checkpoint E.13 runs only where an assistant contributed a specification, a build path, a layout, a title, alternative text, or a sentence. Both are recorded as not applicable with their reason, never omitted silently.

The third is that a pre-read and a dashboard are different objects and the checklist treats them differently, which is the operational form of Section 12.2's distinction. On an analytical artifact the checks protect density and inspectability: a legend is acceptable, a footer on every view is required, and an assertion title is out of place. On an explanatory artifact the checks protect hierarchy and speed of comprehension: the footer consolidates to one source band, direct labels replace the legend, and every title is a sentence carrying an obligation. Where a check differs between the two, it says so.

The fourth is the one students most often get wrong. The checks run on the artifact as it will be *delivered*, not as it exists in the workbook. A view that passes every check inside Tableau Desktop and fails check 14.2 when exported to PDF has failed, because the PDF is what the reader sees.

E.4 The Master Checklist

Checkpoint E.1 — Decision, Audience, and Artifact Type

Owned by. Sections 12.1 and 12.2 for the two jobs of a visual artifact; Sections 13.2 and 13.3 for the reader and the form; Table 13.1 for the choice among dashboard, report, data story, and memo. The frame point of Table 8.5 transfers here as the question the artifact was built to answer.

Runs on. Any artifact that leaves the analyst's screen. On a working notebook chart, checks 1.1 and 1.3 alone.

Table E.5

Checkpoint E.1 — decision, audience, and artifact type

ID	Level	The check
1.1	1	The decision the artifact supports is written in one sentence naming the decision-maker, the choice, and the deadline. If the sentence needs an "and," the artifact is being asked to support two decisions and one of them belongs elsewhere.
1.2	2	The primary reader is named, together with how long she will spend, what she already knows, and what evidence would change her mind.
1.3	1	The artifact's dominant job is declared — analytical or explanatory — and the design matches the declaration rather than contradicting it.
1.4	2	The form is the right one for that decision and that reader: chart, table, pre-read, dashboard, report, data story, or memo. A one-time decision made in a meeting by a reader new to the material is a data story with a memo attached, whatever was requested.
1.5	1	The intended three-second takeaway is written down and sealed before any test is run. This is the same obligation as check 12.1 and is recorded once.
1.6	2	The action or next question that should follow the reading is stated.
1.7	1	An explanatory artifact was rebuilt rather than inherited. A view carried over from the analytical version keeps every default the analytical version was entitled to and the explanatory version is not.
1.8	2	Where the artifact serves two audiences at once, the dominant job is named rather than averaged, and the checks that follow are run against that job.
1.9	2	An analytical artifact circulated to other readers carries a written reading guide naming the question each view answers, the grain of one mark in each, and the window they share, so that a reader who disagrees knows what she is disagreeing with.

Most common failure. The exploratory view — made in five minutes, full of defaults, correct for its purpose — pasted into a deck because it was already there. Its close relative is the first draft of Section 13.1: eight correct views of equal size in a regular grid, which asserts that the eight matter equally, a claim the analyst does not believe and would not defend if asked.

If it fails. Failures here are structural and cannot be repaired downstream. A well-built object of the wrong kind is not improved by better titles, and an artifact with no declared job will fail Checkpoint E.8 on hierarchy no matter how carefully its individual views are drawn. Return to Section 13.3, choose the form from the decision and the reader, and rebuild.

Checkpoint E.2 — Data, Source, Grain, and Reconciliation

Owned by. Section 3.3 for grain, Section 12.10 for aggregation in the view, Lab 12.1 Part A for reconciliation, and V1 of Table 12.5, which this checkpoint implements. This is the checkpoint on which an artifact is most often quietly wrong.

Runs on. Every artifact, without exception and without abbreviation.

Table E.6

Checkpoint E.2 — data, source, grain, and reconciliation

ID	Level	The check
2.1	1	The source table or certified extract is named on the artifact, not only in the analyst's memory.
2.2	1	The date window is stated, and its start is defensible rather than inherited from a filter nobody set.
2.3	1	Every filter and exclusion, including defaults, is stated in words on the artifact. A control showing a filter's state is a fact about an interface; a caption stating it is a fact about the evidence.
2.4	1	What one mark represents was written down before the view was inspected, and the view's dimensions and the measure's aggregation produce exactly that.
2.5	1	Every rate names its denominator, and the denominator is the eligible base rather than the convenient one.
2.6	1	Any per-order, per-customer, or per-store metric has been verified by hand on one group. Average line value and average order value differ by a factor no software will warn you about.
2.7	1	Totals reconcile to the certified source wherever reconciliation is expected, and any filtered total that no longer matches carries a stated filter.
2.8	1	Distinct counts use the correct key.
2.9	1	A join did not duplicate records: pre- and post-join row counts were compared and the cardinality was enforced.
2.10	1	Absent observations and zero observations are distinguished. A week in which a store was open and sold nothing is a zero; a week before it opened is a null; a line drawn straight across either one is a claim the data does not make.
2.11	1	Missing values were not silently omitted in a way that changes the claim.
2.12	1	A regenerable view has not been replaced by a stale screenshot where current data is required, and the refresh date is visible.
2.13	1	Any reference value is entered as a constant from a certified source rather than computed from the displayed marks. A pooled rate is computed on the pooled denominator and is not the unweighted mean of the displayed rates; a reference computed from the view also moves when the view is filtered, which makes the sentence above it false.
2.14	2	Where a reference legitimately is a descriptive summary of the plotted marks — a median, a mean, a prior period — it is labeled as one, so that a reader knows the line summarizes the marks rather than standing independently of them.
2.15	1	Where an artifact carries figures describing different populations, each figure's population is named beside it and no figure is used as though it described both.
2.16	1	Every derived date field is reproducible: the window is anchored to the extract rather than to the system clock, and the week-start convention is declared rather than inherited from a data source's configuration.

Most common failure. The grain slip, and it is the most expensive error in Part III because it is invisible. StyleCraft's entire suburban finding disappears into a three percent difference when average order value is computed from order-line rows and reappears as a sixty-six percent difference when it is computed at order grain, and there is no error message, no warning, and no visual difference between the correct chart and the incorrect one. Its companion is the mislabeled claim: a chart built on line-grain data is not wrong for being line-grain, it is wrong when the chart or the text around it makes a per-order, per-customer, or per-store claim the grain does not support.

If it fails. Stop. A Level 1 failure in this checkpoint means the artifact answers a different question from the one it says it answers, and no amount of design repairs that. Fix the grain, the denominator, or the reconciliation, then re-run Checkpoints E.3 through E.6, because the numbers in every title have changed.

Checkpoint E.3 — Question, Comparison, and Chart Form

Owned by. Section 12.5 and Table 12.1 for purpose-first selection (Shneiderman, 1996; Munzner, 2014); Section 12.4 for why the choice is consequential; V3 of Table 12.5.

Runs on. Every view, in every artifact.

Table E.7

Checkpoint E.3 — question, comparison, and chart form

ID	Level	The check
3.1	1	The question the view answers is written in one sentence.
3.2	1	The comparison the reader must make is explicit, and it is the comparison the question requires.
3.3	2	The analytic purpose is named — distribution, comparison, trend, relationship, composition, or geographic — before the form was chosen.
3.4	2	The chart form makes that comparison easy, and the wrong-form column of Table 12.1 has been read for this purpose.
3.5	2	A table is used where precise lookup matters more than visual pattern.
3.6	2	Categories are sorted in the order the comparison needs, and the sort was chosen rather than left at the software's default.
3.7	1	Time runs chronologically, and a line is not drawn across categorical positions, which invents a continuity the data does not have.
3.8	2	Where a time series is drawn as bars, discrete periods are the intended reading. Bars are appropriate for discrete monthly or annual comparisons and discard the continuity the eye reads best when the movement between periods is the finding.
3.9	2	Composition is not shown in a form whose middle categories cannot be ranked, and shares are not compared across segments whose baselines float.
3.10	2	Small multiples were considered wherever an aggregate could hide heterogeneity, and the panel order is reproducible from a field rather than arranged by hand.
3.11	1	The view answers the question it was built for, rather than a different question the software aggregated into.
3.12	1	The units placed on a common scale are comparable. A shared axis is an implicit claim that the things it holds can be compared, and that claim is made by the layout rather than by the data.
3.13	2	Where a purpose required more than one view, the additional views were built rather than the purpose narrowed to fit one.
3.14	1	Where a view carries a claim — an assertion title, a finding in accompanying prose, a recommendation — the question it answers was declared before the view was built. Exploratory views legitimately generate questions rather than answer declared ones; what is not available is producing several views, selecting the one that says something, and presenting it as though its question had been asked in advance, which is the visual form of the multiple-comparisons problem of Section 11.8.
3.15	2	Where a view bins a continuous variable, the bin width was chosen and stated. Binning is a transformation applied before encoding, so it changes the picture without changing the data.

Most common failure. Check 3.12, and it is the one that reverses recommendations. Every aggregate view of StyleCraft's newest stores supports the sentence "the Wave 4 stores are underperforming," and the comparison being made — eight stores open four months against nine stores open one to two years — is one no analyst would make in a table and that the chart makes irresistible, because the eye compares whatever is placed side by side on a shared scale and does not ask how old anything is. Change the horizontal axis from calendar time to store age and the recommendation flips, with no filter applied, no row removed, and no measure redefined.

If it fails. Return to Table 12.1 and select from the purpose rather than from the form. Where 3.12 fails, the repair is a different horizontal variable, a facet, or an explicit statement of what the comparison does not control for — and the third option is a legitimate answer only when it is written on the chart.

Checkpoint E.4 — Marks and Encodings

Owned by. Section 12.3 for the grammar, Section 12.4 for the perceptual hierarchy (Cleveland & McGill, 1984; Heer & Bostock, 2010), Section 12.11 for faceting.

Runs on. Every view.

Table E.8

Checkpoint E.4 — marks and encodings

ID	Level	The check
4.1	2	The mark type's implied reading is the intended one: bars read as discrete comparisons, lines as continuous trends.
4.2	1	The most important comparison is carried by the highest-ranking channel practical — position on a common scale, then position on identical non-aligned scales, then length.
4.3	1	Angle, area, volume, and color intensity are not carrying a magnitude the reader must decode precisely.
4.4	1	Size encodings scale area rather than radius, map zero to zero area, and carry a size legend with labeled reference values.
4.5	2	Shape encodings remain distinguishable at the size the artifact will actually be seen.
4.6	2	Multiple encodings do not contradict one another, and no field is encoded twice without a reason that can be stated.
4.7	1	Three-dimensional effects are absent. There is no analytical case for them.
4.8	2	Legend categories match the field's level of measurement, and direct labels replace the legend wherever the lookup would cost the reader more than the labels cost the canvas.
4.9	2	Faceting's cost was accepted deliberately: every panel shares one scale, and the trade of one rung of the hierarchy for the ability to see many subsets at once was the right one for this question.
4.10	1	Where a magnitude comparison and a spatial question were both wanted, they were split across two views rather than forced into one. A map answers where; it does not answer how much.
4.11	1	A filled or shaded map encodes a rate over a defined area rather than a raw count. A choropleth of counts mostly draws a population map, and a map of a per-store quantity mostly draws where the stores are.

Most common failure. Spending the best channel on the least interesting fact. A chart in which every category is a different saturated hue has used its most attention-grabbing channel to encode something the axis already said, and a chart that puts the decision's magnitude comparison into circle area on a map has spent its worst channel on its most important question.

If it fails. Name the comparison, name the channel carrying it, name that channel's rank, and reassign. The repair is usually a different mark rather than a different palette.

Checkpoint E.5 — Axes, Scales, and Baselines

Owned by. Section 12.6 and Table 12.2 for the obligation by chart form; Section 12.7 and Table 12.3 for the mechanics; Section 12.9 for references (Few, 2012; Tableau, 2026d); the *baseline* point of Table 8.5 and V2 of Table 12.5.

Runs on. Every view. On a working notebook chart this checkpoint and Checkpoint E.2 are the minimum.

Table E.9

Checkpoint E.5 — axes, scales, and baselines

ID	Level	The check
5.1	1	Ordinary magnitude bars begin at zero. A bar encodes value in length and length is read as a proportion, so a truncated bar does not exaggerate, it makes a false statement in the channel readers trust most.
5.2	1	Floating, interval, Gantt, and waterfall bars begin at their explicitly defined reference, and the reference is stated on the chart.
5.3	1	Stacked bars and areas begin at zero, and it is stated whether the segments are values or shares.

ID	Level	The check
5.4	1	Every position-encoded axis range was chosen deliberately and is stated where a reader might be misled. A restricted line-chart axis is honest only if the restriction is visible and defensible; “the software chose it” is not a defense.
5.5	2	A logarithmic scale is used only where the question is multiplicative — a growth rate, a ratio, a proportional difference, or a comparison across orders of magnitude — every plotted value is positive, and the scale type is noted for a reader who will not check it.
5.6	1	Axis units are visible, and the aggregation is named where the number is a sum, an average, or a rate.
5.7	1	Panels meant to be compared carry identical scales, and any dual axis in a comparison view is synchronized.
5.8	1	Dual axes are absent unless the two scales are fixed by definition. Two independent ranges manufacture the coupling the reader will read as a relationship.
5.9	1	The displayed window covers at least one full seasonal cycle in a seasonal business, or the shortfall is disclosed and no seasonal claim is made on the strength of it; and the pre-period is shown wherever an event is being credited with what followed it.
5.10	1	The baseline, target, prior period, control, comparison group, or benchmark the reader needs to interpret the values is on the chart. Where there is nothing to compare against, the title contains no comparative word.
5.11	1	Every reference line is labeled, and an externally supplied decision threshold was declared before the results were inspected. A threshold chosen after the fact is a description, not a benchmark.
5.12	2	A decision threshold, a forecast trigger, or an experimental cut is not presented as a neutral constant.
5.13	1	Every axis was re-read after the last change to the view. Ranges recompute silently when a filter is applied, a measure is swapped, or a table calculation is added.
5.14	1	An index or percent-change view draws its base — 100 or zero percent — as a labeled reference line, and names the base period explicitly.

Most common failure. Axis truncation, and the reason it needs a Level 1 check rather than a reminder is that the distortion survives everything a reader can do about it. Drawn on an axis running from $240,000 to $255,000, StyleCraft’s quarterly move from $241,000 to $252,000 — a real increase of 4.6 percent — produces a second bar roughly twelve times the height of the first, so that a reader converting the drawing into growth sees something near 1,100 percent. The axis labels are correct throughout. Experimental work finds that the distortion persists when the axis is shown clearly and persists when readers are warned (Pandey et al., 2015; Correll et al., 2020), which is why “the numbers are right there” is not a defense and why the repair has to be structural.

If it fails. Draw length-encoded marks from zero and look at what remains. If the honest rendering shows almost nothing, that is the finding, and the decision about whether there is a finding at all is analysis rather than design. Where a small movement is genuinely the point, move it to a position-encoded view — a line on a restricted, labeled axis, or a percent-change chart drawn from a zero reference — rather than restricting a length.

Checkpoint E.6 — Titles, Labels, Annotation, and Footers

Owned by. Section 12.9 for annotation and the footer discipline; Section 13.6 and Table 13.3 for assertion titles; Section 11.10 for the verbs; C1 of Table 13.6, which this checkpoint implements.

Runs on. Every artifact. On an explanatory artifact this checkpoint is run sentence by sentence and the count is part of the record.

Table E.10

Checkpoint E.6 — titles, labels, annotation, and footers

ID	Level	The check
6.1	2	Analytical views carry descriptive titles appropriate to inquiry; explanatory views carry assertion titles, and only where the assertion is fully supported.

ID	Level	The check
6.2	1	Every assertion title is verifiable from the marks in the view directly beneath it, in that view's own units, with no other evidence admitted. Cover what you know, look only at the marks, and confirm a reader could do it. Every number in an assertion title is therefore visible in the view or directly derivable from what is visible.
6.3	1	The verb is one the evidence licenses. *Rose*, *is*, *reached*, *differs*, and *is associated with* are descriptive verbs a view can support; *drives*, *causes*, *proves*, and *will* are claims about mechanism or the future that a descriptive view cannot carry however strong the analysis behind it.
6.4	1	Every number in accompanying prose — a subtitle, an annotation, a source-band line, a memo sentence — appears in a view or in the certified metrics file the views are built from, and is traceable to whichever it came from. The stricter rule of check 6.2 governs assertion titles, which a reader verifies against one view and cannot check against a file she does not have.
6.5	1	The population in the sentence is the population in the view. Four stores are four stores and not a format; a statement about physical stores is not a statement about the chain.
6.6	1	Percentage change and percentage-point change are distinguished wherever both are available.
6.7	1	Units appear in the axis, the labels, or the title, and every abbreviation is defined.
6.8	2	Labels are direct wherever they remove a legend lookup, and the labeled set is small enough to remain readable.
6.9	2	Annotation explains a relevant event, break, threshold, or exception rather than decorating the view, and the number of annotations is small enough that they retain their force.
6.10	1	The footer carries source, window, filters, grain, and refresh date whenever the artifact leaves the workbook — under every view on an analytical artifact, once for the screen on a dashboard, with per-view grain moved into tooltips only where nothing load-bearing goes with it.
6.11	1	A metric definition accompanies any number a reader could otherwise misinterpret.
6.12	1	A title over a parameterized or filterable view remains true in every state the control can produce, or the title is built dynamically so that it tracks the control.
6.13	2	The claim-making sentences were counted — every title, subtitle, annotation, source-band line, and line of accompanying prose — and that count is the denominator reported against check 6.2.

Most common failure. The overclaiming title, in three recognizable shapes: a causal or predictive verb substituted for a descriptive one, a population generalized from the observations to the category, and a number imported from somewhere other than the view. All three are more interesting sentences than the true one, which is why they are produced and why they survive review.

If it fails. Reject the sentence rather than repairing it, because repairing a title reintroduces the analyst's own knowledge into a sentence that has to stand without it. Then notice what the true sentence obliges: the moment a title says "seven of the eight," the view acquires a duty to show eight things distinguishably, and the title has forced a design change rather than labeling one. That is the technique working in the productive direction.

Checkpoint E.7 — Color and Accessibility

Owned by. Section 12.8 and Table 12.4 for palette type by level of measurement; Section 13.7 for emphasis as a budget and for data-ink discipline (Tufte, 2001; Few, 2012); W3C (2024) for the accessibility requirements the guide adopts as conditions of correctness.

Runs on. Every view. Checks 7.1 through 7.5 are Level 1 in every artifact, including a working notebook chart that will be submitted.

Table E.11

Checkpoint E.7 — color and accessibility

ID	Level	The check
7.1	1	Palette type matches the field's level of measurement: categorical for unordered groups, sequential for ordered ones and for quantities, diverging only where a meaningful midpoint exists and both distance and direction from it are the point.
7.2	1	Color is not the only visual means by which any essential distinction is conveyed. Position, ordering, direct labels, shape, line pattern, or lightness carries it as well.
7.3	1	The grayscale screening has been run: convert the artifact and confirm the distinctions survive. Collapse means rebuild; survival means one test passed rather than all of them, because grayscale exposes reliance on hue and weak lightness contrast and does not simulate every deficiency.
7.4	1	The artifact has been inspected through a color-vision-deficiency simulator. Roughly one in twelve men of Northern European descent has some form of red–green deficiency, and the deficiency is invisible to everyone else in the room including the person who chose the palette.
7.5	1	Contrast meets a stated criterion at the condition the artifact will actually be read in — a projector in a lit room, a phone, a printed page — rather than on the analyst's monitor. This guide adopts WCAG 2.2 Success Criterion 1.4.3 for text, at 4.5 to 1 against its background, and 1.4.11 for graphical objects and interface components, at 3 to 1 against adjacent colors (W3C, 2024).
7.6	2	Red-to-green is not used as an unexamined good/bad shorthand, and no unordered rainbow scale carries an ordered quantity. An ordered quantity takes a perceptually ordered palette — sequential where it has no meaningful center, diverging where it does — chosen for the variable and for the comparison the reader must make.
7.7	2	Neutral categories are not assigned unintended positive or negative valence.
7.8	2	Emphasis color is spent rather than spread: one accent for the subject, gray for the context, and an emphasized set small enough that the reader does not have to search inside it.
7.9	2	Background, gridlines, borders, and decoration are de-emphasized or removed, and every retained non-data element is doing a job that can be named. Reference lines, direct labels, annotation, and the footer are non-data ink and are kept.
7.10	2	A brand palette is used for identity and accent, with any sequential ramp derived by varying lightness within one brand hue and a neutral gray as the workhorse. A diverging palette takes two accessible hues around a neutral midpoint, since it cannot be built inside a single hue. Where a brand guideline and an encoding requirement conflict on an analytical artifact, the encoding wins.
7.11	2	A categorical palette carries no more than seven or eight hues. Beyond that, color alone stops being a reliable identifier and the distinction moves to direct labels or to another channel.

Most common failure. An ordered field in a categorical palette, or an unordered field in a sequential ramp, with an accessibility omission alongside it — because nothing in a request for a chart mentions color-vision deficiency and no software default raises it. StyleCraft's four loyalty tiers in a red-to-green ramp fail three ways at once: they use the two hues most often confused, they carry the order in hue rather than in lightness, and they assert a valence the tiers do not have.

If it fails. Rebuild the palette from the field's level of measurement rather than adjusting hues, then re-run 7.3 and 7.4. Where color was carrying an essential distinction alone, the general repair is redundant encoding, and direct labels, position, and ordering are better safeguards than any choice of hue.

Checkpoint E.8 — Layout and Visual Hierarchy

Owned by. Sections 13.4 and 13.5 for the KPI hierarchy and its spatial expression (Few, 2006); Section 13.7 for decluttering as the precondition of emphasis (Knaflic, 2015); Treisman and Gelade (1980) and Healey and Enns (2012) for why emphasis is a budget.

Runs on. Any artifact carrying more than one view — a dashboard, a report page, a pre-read spread.

Table E.12

Checkpoint E.8 — layout and visual hierarchy

ID	Level	The check
8.1	1	The KPI hierarchy was written down before the canvas was opened: one or two primary views, a supporting tier, a diagnostic tier, each defined by the reader's question it answers.
8.2	1	Prominence matches the hierarchy. The view the decision turns on is the largest object and sits on the reader's entry path, even when it is the plainer object.
8.3	2	Position and prominence agree rather than fight; a large high-contrast object elsewhere on the canvas does not capture the entry fixation.
8.4	2	Reading order matches the sequence in which the reader must acquire the argument.
8.5	2	Views in the same tier are adjacent and share their formatting; views sharing a scale are aligned, and their axes were confirmed identical rather than assumed.
8.6	1	The number of findings the reader must hold at once was stated before layout and is small. Everything else is available in sequence rather than simultaneously.
8.7	2	Whitespace separates functions rather than filling unused space, and padding was set rather than left at zero.
8.8	1	Every view can be justified by its tier and by the point in the reader's sequence at which she needs it. A view with no justification comes off the canvas and goes into the memo's appendix; it does not become a smaller view.
8.9	1	The case's weakest facts are on the canvas rather than in a backup. A reader who discovers a weakness herself concludes that the analyst either missed it or hid it, and both conclusions are expensive.
8.10	2	The sizing mode matches the consumption condition — fixed at a known projection resolution, device-specific for unknown displays — and the choice was recorded rather than defaulted.
8.11	1	The squint test was run: blur the artifact until the text is unreadable and confirm the intended object still dominates. If it does not, the hierarchy is in the analyst's head and the repair is size and position rather than a bigger title.
8.12	1	The primary message survives at the intended display size, and no essential evidence sits below the fold or behind an unexplained interaction.
8.13	2	The artifact can be read without the analyst narrating every element.
8.14	2	Nothing was solved by shrinking. A canvas of twelve small views is not a decluttered canvas of twelve views; it is an illegible one.
8.15	2	Where the authoring tool offers layout containers — as Tableau does — the canvas is built from nested containers rather than floating objects, so that the tiers hold their relationships when the artifact is resized or something is added late. In a tool without them, the equivalent obligation is that the layout survives a change of size or a late addition without being rebuilt.
8.16	2	The title band carries the decision the screen supports and the date rather than a restatement of the data, and the analyst of record is named on it.

Most common failure. Demoting instead of cutting. The tier structure creates a temptation to solve a crowded canvas by making things smaller, and the result restores the undifferentiated grid at a lower point size. Its companion failure is the emphasized set that has grown: a canvas emphasizing six things has emphasized nothing, because the reader must now search the emphasized set.

If it fails. Declutter against the artifact's own sentence rather than against a general principle. With the assertion title written down, every element can be asked whether it helps a reader believe that specific sentence, and elements with no defensible answer come off without further argument. Move rather than delete — a view to the memo's appendix, a filter to the analytical tab, a footnote to a tooltip — so that nothing is lost and the reader's ten minutes are recovered.

Checkpoint E.9 — Interactivity

Owned by. Section 13.8 and Table 13.5 for what communicates and what conceals; Shneiderman (1996) and Heer and Shneiderman (2012) for the interaction types; Tableau (2026a, 2026c) for the mechanics of actions and filter scope.

Runs on. Any artifact a reader can change: a dashboard, a published workbook, an interactive report.

Table E.13

Checkpoint E.9 — interactivity

ID	Level	The check
9.1	1	Every finding the artifact must communicate is visible in the default state. Interactivity may add depth to a finding that has already landed; it may not deliver one.
9.2	1	Each control exists to answer a reader question the analyst can name.
9.3	1	The state the reader meets is disclosed in words on the canvas rather than shown only in a control. Where a screen that could carry controls carries none, it says so, because a screen with nothing to adjust otherwise makes no claim about its own completeness.
9.4	1	No control can falsify a sentence on the same screen. Where no safe scope exists, the control comes off and the state is declared in words instead.
9.5	1	Filter scope was chosen deliberately and stated. A filter applied to every worksheet on a data source will move views whose titles quote chain-wide figures and whose references were computed on another population.
9.6	1	A filter written against a field that only one kind of row possesses does not silently delete the other kind.
9.7	1	No filter or action combination produces an empty, partial, or nonsensical state without explanation — a view emptied of its marks, a frame whose count no longer matches the sentence above it, or a region deleted while its caption remains.
9.8	2	The reader can return to the default state, and the route back is discoverable.
9.9	1	Cross-filtering does not silently change a denominator.
9.10	2	Where one view's selection drives another — a Tableau highlight action, a cross-filter, a linked selection — the interaction runs from the smaller value set to the larger, every target contains the values the source passes it, and each was tested and observed to fire. An interaction that does nothing reads as a broken artifact rather than as an empty match.
9.11	1	Nothing load-bearing lives in a tooltip. A qualification that makes a headline defensible belongs on the canvas.
9.12	2	Navigation actions carry a visible cue that the deeper view exists.
9.13	1	The static export still contains the essential claim and its context.
9.14	2	The interaction inventory is complete: for every interaction, the reader question it answers, what a reader who never triggers it misses, and confirmation that it fires. Where the answer to the second names a finding, check 9.1 has failed and the finding moves onto the canvas.

Most common failure. The presence of many controls, which is itself a message, and the message is "this is a tool for you to use" — the wrong message for a ten-minute reader making a one-time decision. Underneath it sits the assumption that defeats most dashboards: the analyst who built the artifact will click everything, a colleague will click a little, and an executive in a meeting will click nothing, so the analyst reliably overestimates how much of her design the audience will ever see.

If it fails. Apply the first row of Table 13.5 honestly to the specific artifact. A filter communicates when it narrows a population the reader already understands *and* cannot falsify anything on the screen; where the second condition cannot be met at any scope, the honest number of controls is zero, and the state moves into the source band. Removing a control is a communication decision rather than a simplification, and it belongs in the specification with its reasoning.

Checkpoint E.10 — Uncertainty and Decision Conditions

Owned by. Section 13.9 for the executive treatment; Chapters 10 and 11 for the intervals themselves; Section 2.7 for asymmetric cost; the *error in decision units* point of Table 8.5 applied to a range rather than to a point. Correll and Gleicher (2014) and Spiegelhalter et al. (2011) supply the encoding evidence, and Tableau (2026d) is why check 10.6 cannot be satisfied with a reference band.

Runs on. Any artifact carrying a forecast, an estimated effect, a threshold, or a recommendation.

Table E.14

Checkpoint E.10 — uncertainty and decision conditions

ID	Level	The check
10.1	1	Material uncertainty is present wherever the decision depends on it. A point estimate presented alone transfers risk from the analyst's credibility to the organization's balance sheet.
10.2	1	The interval is named correctly — prediction interval, confidence interval, simulated range — and its level is stated.
10.3	1	The interval is expressed in decision units, and the conversion is shown on the canvas in steps with its denominator named rather than asserted as a result.
10.4	1	Where a conversion would require an assumption that cannot be established, what is measured or forecast was changed so the assumption is not needed; where that is impossible, the assumption is disclosed where a reader will find it rather than buried in an arithmetic step.
10.5	1	Where both framings are used, both are given: the end-to-end width of the range and the spread around the central estimate. Quoting the first alone overstates how uncertain the estimate is.
10.6	1	Interval bounds exist as values at the display's own grain, computed upstream rather than requested from the drawing tool. A region whose edges move across the horizontal axis is a shape built from those values — in Tableau a polygon over computed vertices, since a reference band renders at constant height; in a plotting library, a fill between two series.
10.7	1	An interval over an aggregate is not produced by summing the bounds of its component periods, which overstates the width. Where the interval comes from a simulation, the aggregate is formed on each path and the quantiles are taken from the aggregated totals; where it comes from another method, the aggregate interval is derived by that method's own rule and the rule is named.
10.8	1	Periods that have already resolved are shown as actuals outside the interval, with the seam marked and the forecast origin stated. Presenting elapsed months inside a band prices risk that no longer exists.
10.9	2	False precision has been removed and every assumption is named.
10.10	1	The range is not visually compressed — by an over-wide axis, a long empty tail, or a scale chosen after the interval was known — so as to appear more certain than it is.
10.11	1	The artifact does not imply that an observational relationship is an experimental effect.
10.12	1	The complete decision artifact carries all four parts: the recommendation, the evidence, the uncertainty, and the conditions under which the recommendation would change.
10.13	2	Where the cost of being wrong is asymmetric, the asymmetry is stated in comparable units, with both subtractions shown rather than two figures set against each other that are not comparable.
10.14	1	Every assumption supplied rather than measured — a planning figure, a cost, a capacity — is attributed to its source on the artifact rather than presented as something the data produced.
10.15	1	A recommendation states the rule before any count, and any count it carries is reconcilable to a table the reader can be shown. A count that cannot be reproduced from a stated rule is an opinion with a number attached.
10.16	2	The reversal conditions are a small, enumerable set — two or three is the working shape — each with a date or a threshold attached, and at least one names what would be re-estimated when the underlying data next refreshes.
10.17	2	Where part of a forecast horizon has already resolved, the model's prediction for those periods was compared against what was reported, and the comparison is disclosed as evidence for or against the range on the periods still open.

Most common failure. The interval that disappears during editing, and it is worth predicting explicitly because it is so reliable: asked to make a paragraph clearer or punchier, an assistant removes the interval before it removes anything else, because the interval is the clause that complicates the sentence. The human version is the same move made by the analyst, on the grounds that a range will be read as incompetence.

If it fails. The alternative to a hedge is not confidence; it is the four-part shape — here is what I recommend, here is the evidence it rests on, here is the range I cannot narrow in units you care about, and here is what would have to be observed for me to change the recommendation. The fourth part is what converts an interval into a monitoring plan, and it is what makes a decision made under uncertainty cheap to unmake.

Checkpoint E.11 — Persuasion, Omission, and Ethical Review

Owned by. Section 12.16 for the disclosure standard; Section 13.15 for the boundary between emphasis and distortion; C3 of Table 13.6, which this checkpoint implements; Hullman and Diakopoulos (2011) for the framing effects, Tversky and Kahneman (1981) for the framing result.

Runs on. Any artifact built for an audience — which is every artifact that is not a private working chart.

Table E.15

Checkpoint E.11 — persuasion, omission, and ethical review

ID	Level	The check
11.1	1	The counterfactual has been asked and answered in writing: would this reader revise her conclusion if she saw everything the analyst saw? A yes means the compression or the emphasis requires repair or disclosure.
11.2	1	Every removal made for the audience is listed — views cut, categories grouped, windows shortened, filters defaulted or scoped, qualifications moved into tooltips — and each is sorted into a legitimate simplification or the deletion of a qualification.
11.3	1	Grouping into "other" does not hide a category whose individual value would change a ranking or a conclusion.
11.4	1	The quantity is presented in a form that does not select the reader's reaction, and an equally true alternative framing was considered rather than avoided.
11.5	1	The views shown represent the evidence rather than the favorable part of it, and inconvenient findings remain available or are disclosed.
11.6	1	A title does not select the favorable reading where the same view equally supports an unfavorable one that the decision turns on.
11.7	1	Axis, canvas, and scale choices do not minimize uncertainty or adverse outcomes.
11.8	2	The artifact would survive its own audience being fully informed: hand the reader the full evidence and an afternoon, and expect the same decision. An artifact that depends on the reader not having that afternoon is an argument requiring an uninformed audience.
11.9	2	The standard applied was *did I check*, which leaves evidence, rather than *did I intend*, which cannot be audited by anyone.
11.10	2	Emphasis was not abandoned in the name of neutrality. A screen with no hierarchy is not a neutral screen; it is one whose hierarchy was set by the order the views were built in, which still produces a conclusion in the reader's head and is unaccountable for it.
11.11	2	The views that were built and rejected are listed, with one line apiece on why, so that a reader can see the analyst's search rather than only its result.
11.12	1	A case singled out for attention was selected by a rule declared in advance rather than chosen after inspection and then explained, which is post hoc hunting in a different costume.

Most common failure. The same sentence — "it was cluttering the screen" — is available to describe a view removed because the memo carries it better and a view removed because it complicates the recommendation. That is precisely why check 11.2 asks for the sorting to be done deliberately and in writing rather than left to the analyst's sense of her own motives.

If it fails. Anything that survives the counterfactual with a yes goes back on the artifact, at whatever size and however inconvenient. Where the material genuinely will not fit, it goes into the memo's appendix and the artifact says on its face that it is there.

CONCEPT

Emphasis Directs Attention; Distortion Directs Belief

Section 13.15 draws the distinction this checkpoint enforces and is where the argument belongs; what the checklist adds is the order in which to run it. Ask check 11.1 first, because it is the only test that reaches a defect assembled entirely out of accurate elements. Then run 11.3, 11.4, and 11.5, which are the three moves — aggregation, framing, and selection — through which that assembly is actually performed.

The reason the order matters is that 11.3 through 11.5 can each be answered yes in good conscience while 11.1 is answered no, and the reverse never happens.

Source: Course concept developed for this guide, informed by Section 13.15, Hullman and Diakopoulos (2011), and Tversky and Kahneman (1981).

Checkpoint E.12 — Usability Testing

Owned by. Section 13.2 for the three-second test as a course heuristic; Section 13.10 for the protocol and the critique exchange; C2 of Table 13.6. Krug (2014) and Doncaster (2014) supply the technique; Nielsen and Landauer (1993) supply the caution about how much a small test establishes.

Runs on. Explanatory dashboards, decision artifacts, and major project deliverables. Not on a working notebook chart, and not on a pre-read whose readers are the people who will interrogate it.

Table E.16

Checkpoint E.12 — usability testing

ID	Level	The check
12.1	1	The intended takeaway was written down and sealed before the test. A takeaway written afterward is not a prediction and the protocol has no value without one.
12.2	2	Three further predictions were sealed: what the reader's eye lands on first, the first question she asks, and the element most likely to be misread.
12.3	1	The reader is representative and has not seen the artifact. A colleague who built the same thing last week knows where to look and is a test of nothing.
12.4	1	The artifact was shown for approximately three seconds, removed, and the reader's takeaway recorded verbatim rather than paraphrased.
12.5	1	A ninety-second task followed — "tell me what you would decide and why" — with reading order, questions, hesitations, and the stated decision recorded.
12.6	1	The analyst said nothing during the task: no explanation, no guidance, no answering of questions until the time was up.
12.7	2	A mismatch in the *subject* of the reader's sentence was investigated as a hierarchy failure first, since size and position are the cheapest repair and the most common cause.
12.8	2	A mismatch in the *verb* was investigated as a title or emphasis failure first, and repaired in language and color before anything structural was changed.
12.9	2	Views the reader skipped entirely were distinguished from views that failed to persuade. A skipped view was never read, and the two have different repairs.
12.10	1	Material mismatches produced a revision and a retest on a fresh reader; accepted defects were recorded with one sentence of reason each.

ID	Level	The check
12.11	1	The result is reported as a formative check on one or two readers, which finds gross hierarchy and language failures, rather than as validation that the artifact works for its actual audience.
12.12	2	The memo alone, without the artifact, was handed to a third reader who could state the recommendation and the basis for it.
12.13	2	Where a peer critique was run, it followed the protocol in order — report the apparent takeaway, ask what decision the artifact supports, identify the specific elements that impede reading, and only then propose changes — with the author listening without defending until the sequence is complete.
12.14	2	Every piece of feedback declined is recorded with one sentence of reason, alongside the changes that were made and the evidence that prompted each.

Most common failure. Talking. The reflex to explain a view the moment the reader misreads it destroys the measurement and is nearly irresistible, which is why check 12.1 exists: writing the prediction down first is what puts something at stake in staying quiet.

If it fails. Revise the artifact, not the reader. The reader's response is the measurement rather than the error, and a reader who says the artifact is about the new stores doing badly has told you something true about your hierarchy that no amount of inspection would have surfaced.

Checkpoint E.13 — AI-Drafted Visual or Communication Audit

Owned by. Section 12.12 for the chart-drafting failure modes and the five-step routine; Section 13.11 for the language failure modes; Appendix C, Templates C.16 through C.18, for the prompts; Appendix D for the record.

Runs on. Any artifact whose specification, build path, layout, title, alt text, or accompanying prose an assistant contributed to. Run it in addition to the other checkpoints, never instead of them.

Table E.17

Checkpoint E.13 — AI-drafted visual or communication audit

ID	Level	The check
13.1	1	The faults were predicted from the specification, in writing, before the view was rendered. A rendered chart is persuasive in a way its specification is not, and the human tendency is to evaluate whether it is attractive rather than whether it is right.
13.2	1	The specification was built exactly as written, faults included, so that the faults can be audited rather than corrected invisibly. This applies in an instructional or otherwise safe environment; a specification that would create a privacy, security, cost, or irreversible operational exposure is audited on paper and never executed.
13.3	1	No field, filter, value, or business fact was invented. Every column referenced exists in the schema that was supplied.
13.4	1	The grain and the aggregation were stated by the analyst rather than chosen by the assistant.
13.5	1	Software and assistant defaults were inspected individually: axis range, sort order, palette, aggregation, filter state, mark type.
13.6	1	Candidate titles were verified against the built view one sentence at a time and rejected on first failure rather than repaired.
13.7	1	No causal or predictive verb was introduced at the narration step, where assistants reintroduce it even when the numbers are clean.
13.8	1	No interval, threshold, qualification, or reversal condition was removed during a rewrite. The two versions were diffed clause by clause rather than the new one read.
13.9	1	A proposed layout implements the analyst's KPI hierarchy, supplied as an input, rather than a conventional grid the assistant has no basis for.
13.10	1	No connective sentence asserts a relationship between two views that no view establishes. It reads as synthesis; it is confabulation, and only the traceability rule catches it.

ID	Level	The check
13.11	2	Alt text, footer content, contrast, and filter-state disclosure were requested explicitly, because nothing in a request for a headline or a layout will produce any of them unasked.
13.12	1	The prediction was graded: both the faults missed and the faults predicted that did not occur are recorded, since a systematic error in the analyst's model of the assistant is worth more than any single caught title.
13.13	1	The exchange is documented per Appendix D, including the output that was rejected.
13.14	1	The analyst can explain and defend every encoding and every sentence in the artifact. An assistant can write a sentence; only the analyst can sign it.
13.15	1	Every candidate sentence arrived with its own verification line naming the marks a reader would have to look at to confirm it. A candidate whose verification line is vague is a candidate that cannot be verified.
13.16	2	The surviving title was shown on its view to someone who had seen neither, and her paraphrase was the claim the analyst intended rather than the stronger one next door.

Most common failure. Two, one per medium. On a chart, the axis truncated on request: an assistant asked to "make the growth clear" will restrict the axis without comment, because the instruction was followed. On a sentence, the overclaim, because a more interesting sentence is better writing and the assistant is optimizing for writing.

If it fails. Repair one mechanic at a time against the checkpoint that owns it, writing a repair note per Section E.6 for each. Where the failure is a sentence, discard it and take a different candidate rather than editing the one that overclaimed.

Checkpoint E.14 — Export and Delivery

Owned by. Section 12.9 for the footer discipline, Section 13.5 for sizing, and Lab 13.1 Part C for alt text; Tableau (2026b, 2026e) for the mechanics. The delivery checks themselves are this appendix's own: no chapter section owns the export, because in both chapters the artifact is inspected where it is built.

Runs on. Every artifact at the moment it is delivered, in the form it is delivered in.

Table E.18

Checkpoint E.14 — export and delivery

ID	Level	The check
14.1	1	Final dimensions match the delivery channel, and text is readable at actual size in the room, on the page, or on the device where it will be read.
14.2	1	The static export preserves labels, legends, reference lines, annotations, and the footer.
14.3	1	Interactive links open with the intended permissions and in the intended default state.
14.4	2	The workbook, notebook, or query accompanies the artifact wherever reproducibility is required.
14.5	1	The file name and version are unambiguous, and the version on the artifact matches the version in the sign-off record.
14.6	1	The data refresh date is visible wherever recency matters.
14.7	1	Alternative text describes the view's purpose, population, units, main pattern, and any material exception — not the field names, which is what the software supplies by default.
14.8	1	Every recipient is authorized for everything the delivery carries. Labels, tooltips, annotations, file names, and the extract behind the artifact expose no protected field and no more detail than the decision requires, permissions on any interactive link match the intended audience, and nothing travels that the analyst is not authorized to disclose to the people who will receive it.
14.9	2	A decision-supporting artifact travels with its written recommendation rather than alone.
14.10	2	The sign-off record of Section E.7 is complete, and the analyst of record is named on the artifact or travels with it.

Most common failure. The artifact that passed every check inside the authoring tool and lost half of them on the way out: a legend clipped in the PDF, a tooltip carrying the grain statement that the printed copy cannot show, alt text still naming fields, an interactive link that opens for the analyst and not for the recipient.

If it fails. Repair and re-export, then check the export rather than the source. The only version that matters is the one the reader receives.

E.5 Artifact Short Forms

These four forms are the working instrument for every run after the baseline audit, on the rule stated in Section E.3: the first release of an artifact and any graded submission run the applicable master checkpoints in full, and every run after that is a short form. Each form is a front end to the master checklist rather than a substitute for it. Every item consolidates one or more master checks and cites them by identifier, so the two can never disagree: an item is satisfied when its cited checks are satisfied. The forms carry no level column, because the level belongs to the master check — an item citing 5.1 and 5.4 inherits Level 1 from both, and an item citing 3.10 inherits Level 2.

A short form is deliberately shorter than the checkpoints behind it. Its questions consolidate, and consolidation loses resolution: a form asks about axes rather than about each of the fourteen axis checks, and a defect that no cited question would surface is a defect the form will not find. That is the cost of speed and the reason the baseline audit is not optional. On a revision it is the right trade, because the baseline audit has already looked everywhere once and the short form is looking at what changed.

When a short-form item is satisfied on inspection, record it and move on. When it fails, or when its answer is not obvious, open the master checkpoint it cites, work the individual checks, and record the outcome against those identifiers rather than against the short-form question — because a repair note, a re-run, and a chapter cross-reference all need the individual check, and “the axis question failed” names none of them.

E.5.1 Single Chart Sign-Off

For a chart in a working notebook, a figure in a report, or one view built in Tableau. Under two minutes.

Table E.19

Short form E.5.1 — single chart

Ask	From
What decision or question does this view support, and is its job analytical or explanatory?	1.1, 1.3
What question does this view answer, and what comparison must the reader make?	3.1, 3.2
What does one mark represent, and do the dimensions and aggregation produce that?	2.4
Is the channel carrying the main comparison the best one available?	4.2
Does every length-encoded mark begin at zero, and was every axis range chosen and stated?	5.1, 5.4
Is the reference the reader needs on the chart, is it labeled, and was it entered from a certified source?	5.10, 5.11, 2.13
Does the palette match the field, and does the chart survive grayscale?	7.1, 7.3
Does the title claim only what the marks support, with its units and its verb?	6.2, 6.3, 6.7
Does the footer carry source, window, filters, grain, and refresh date?	6.10
Was the sort chosen rather than inherited?	3.6

E.5.2 Analytical Pre-Read Sign-Off

For a set of views built to be argued with. Its governing property is inspectability: a reader who disagrees must be able to see exactly what she is disagreeing with.

Table E.20

Short form E.5.2 — analytical pre-read

Ask	From
Is the artifact's job declared as analytical, and does its density serve inspection rather than speed?	1.3
Is there a written reading guide naming the question each view answers, its grain, and the shared window?	1.9
Does every view state its own grain, window, filters, and source?	2.1, 2.2, 2.3, 2.4, 6.10
Do the totals reconcile, and has one per-unit metric been hand-checked?	2.6, 2.7
Are the units placed on any shared scale genuinely comparable?	3.12
Are the pre-periods, comparison groups, and full seasonal cycles present?	5.9, 5.10
Were small multiples considered wherever an aggregate could hide heterogeneity?	3.10
Are definitions and scales consistent across views, and every reference a certified constant?	2.13, 5.7, 6.11
Are the titles descriptive rather than assertive, so that the reader forms her own conclusion?	6.1
Is the inconvenient view present, and are the charts that were built and rejected listed?	11.5, 11.11
Does the delivered export carry the labels, references, and footers the workbook shows?	14.2, 14.6

E.5.3 Executive Dashboard Sign-Off

For a single-screen decision artifact. This form is the compressed version of the full fourteen checkpoints and does not replace them on a first build; it is what you run on the second and third revisions.

Table E.21

Short form E.5.3 — executive dashboard

Ask	From
Is the decision written in one sentence, with the reader, the choice, and the deadline?	1.1, 1.2
Was every view rebuilt rather than inherited from the analytical version?	1.7
Is the KPI hierarchy written down, and does prominence match it under the squint test?	8.1, 8.2, 8.11
Does the title band carry the decision and the analyst's name?	8.16
Is every sentence on the screen verifiable from the marks beneath it, in its own units, with a licensed verb and no imported number?	6.2, 6.3, 6.4, 6.5
Are the case's weakest facts on the canvas?	8.9
Is the uncertainty present, in decision units, with the conversion shown and its denominator named?	10.1, 10.3
Are the assumptions attributed to their sources, and the reversal conditions stated with dates or thresholds?	10.14, 10.16
Is the default state disclosed in words, and can no control falsify a sentence on the same screen?	9.3, 9.4
Is every finding visible without interaction, and does no control produce an unexplained empty state?	9.1, 9.7
Was the three-second test run on a fresh reader, and did her takeaway match the sealed one?	12.1, 12.4
Is the removal list written and sorted?	11.2
Does the export carry labels, alt text, refresh date, and no restricted data?	14.2, 14.7, 14.8

E.5.4 Recommendation Memo Sign-Off

For the one-page written recommendation that accompanies a decision artifact. A memo is checked against the same sentence-level standard as a title, because it makes the same kind of claim in a more durable form.

Table E.22

Short form E.5.4 — recommendation memo

Ask	From
Does it open with a recommendation — an action or a rule — rather than a finding wearing a recommendation's clothes?	10.12
Is the recommendation a rule before it is a count, and can any count be reconciled to a table in the appendix?	10.15
Does every evidence sentence name the view it comes from and state its finding in decision units?	6.2, 10.3
Is every verb one the evidence licenses, and every population the one the evidence describes?	6.3, 6.5
Do the memo's numbers agree with the artifact's, sentence by sentence?	6.4
Is the uncertainty in the reader's units, with the arithmetic and its denominator shown?	10.3, 10.5
Are the scope limits stated — the populations the evidence does not cover, and the false precision removed?	2.15, 10.9
Is every assumption supplied rather than measured attributed to its source?	10.14
Are there two or three reversal conditions, each with a date or a threshold attached?	10.16
Are the removal list and the Appendix D documentation in the appendix?	11.2, 13.13
Handed to a third reader without the artifact, can she state the recommendation and its basis?	12.12

E.6 The Repair Note

Every Level 1 failure produces a repair, and every repair produces one sentence. The sentence records what changed in the *reading*, not what changed in the chart, and the distinction is the whole point of the instrument: "I set the axis to include zero" is a description of an action, while "a reader now sees a quarter that moved slightly rather than a quarter that appeared to jump twelvefold" is a description of a consequence, and only the second tells anyone whether the repair mattered.

Three uses follow from writing them that way. The note is what goes into the memo you send a colleague whose chart you corrected, where naming the mechanic rather than the person is what makes the message survivable. It is the evidence that a repair was worth making, which matters when a stakeholder asks why an artifact was delayed by a day. And it is the record that lets a later reader distinguish a repaired artifact from one that never needed repair, which is the difference between a checklist that was run and a checklist that was filed.

Table E.23

Repair notes, in the form they should be written

Check	The repair	What changed in the reading
5.1	Redrew the quarterly bars from a zero baseline.	A reader now sees a quarter that moved slightly, rather than a quarter that appeared to jump twelvefold.
5.8	Replaced the dual axis with two panels sharing the time axis, and added the urban series as the comparison group.	A reader now sees two series she must compare deliberately, rather than one apparent relationship she was invited to accept.

Check	The repair	What changed in the reading
3.9	Rebuilt the seven-category pie as a sorted bar, then as a 100 percent stacked bar split by metro.	A reader can now rank the middle categories, which the pie made impossible, and can see the share comparison the pie was pretending to answer.
7.1	Replaced the red-to-green tier palette with a single-hue sequential ramp ordered None to Backstage.	A reader now sees four ordered tiers, rather than two tiers she cannot distinguish and a good/bad judgment the tiers do not carry.
2.4	Repointed the sheet at the order-grain extract and relabeled the measure.	A reader now sees average order value, rather than average line value under a label that promised the first.
6.2	Replaced an assertion title with the sentence the view's marks support.	A reader now takes away a claim she can verify from the chart in front of her, rather than one she must take on trust.

The last column of that table is the deliverable in a way the middle column is not, and it is also the column students find hardest, because writing it requires deciding what the reader believed before and what she believes now. An analyst who cannot state the difference has not established that the repair was necessary.

E.7 The Sign-Off Record

The record is one page and it is the appendix's output. It carries the identity of what was reviewed, evidence that every applicable check was run, the failures found and their disposition, the judgment calls that were accepted, the usability evidence where the artifact required it, and a name. Where the artifact will be submitted for a grade, the record is submitted with it; where it will be circulated at work, the record stays with the workbook.

One design decision in the record is worth stating before the fields. A one-page sign-off cannot carry an outcome line for every applicable check, and it does not try to; the completed short form, or the companion spreadsheet where the master checklist was worked, is where the individual outcomes live. What the record carries instead is the **count line** — how many distinct checks were applicable, how many passed, how many were accepted as Cautions, how many Stops were repaired, and how many remain unresolved — together with the identifier of the completed checklist the counts came from. The counts are in unique check identifiers rather than in check executions, because an artifact of four views runs the same check four times and a one-page record that counted executions would tell a reader nothing she could reconcile. Without that link a sign-off proves that problems were recorded and does not prove that the checks were run, which is a materially weaker claim and the one most sign-off documents in practice actually make.

Table E.24

The sign-off record

Field	What it holds
Artifact title	The name a reader will see, not the file name.
File and version	The delivered file, its version, and the format it was delivered in.
Data refresh date	The date of the extract or the certified source behind the artifact.
Population and window	The population every store-level or unit-level figure describes, and the window, in the words used on the artifact.
Instrument run	Whether this was the baseline audit — the applicable master checkpoints run in full — or a short-form re-run against an existing baseline, and which of the four forms of Section E.5 was used as the front end.
Completed checklist	The file name, spreadsheet tab, or record identifier of the completed checklist the counts below are taken from. A sign-off without this line asserts its counts rather than evidencing them.
Checkpoints run	The checkpoints run, and the checkpoints recorded as not applicable with a three-word reason each.

Field	What it holds
Unique applicable check IDs	How many distinct checks applied to this artifact, given the routing of Section E.3 and the checkpoints marked not applicable. Counted once for the artifact, not once per view: the individual instances live on the completed checklist.
Unique check IDs passed	How many distinct applicable checks were run and satisfied on every instance.
Unique Level 2 check IDs accepted	How many distinct Level 2 checks were accepted as Cautions. Must equal the number of entries in the accepted-defects field.
Unique Level 1 check IDs repaired	How many distinct Level 1 checks failed anywhere in the artifact and were repaired everywhere they failed. Must equal the number of entries in the stop-ship field. One check failing on four views is one identifier, four instances, and potentially four repair notes.
Unique Level 1 check IDs unresolved	How many distinct Level 1 checks failed and were not repaired. Any number above zero means the artifact has not been released; if it was released anyway, the override and the name of the person who authorized it go here.
Stop-ship failures found	Every Level 1 failure by identifier, with its repair note per Section E.6.
Communication revisions	Every Level 2 change made, by identifier.
Accepted defects	Every Level 2 check recorded as Caution, by identifier, with one sentence of reason.
Sentence count	For an explanatory artifact: how many claim-making sentences were checked against 6.2, and how many were changed.
Removal list	The C3 disclosure of check 11.2: what came off, and whether each removal is material to the decision.
Usability evidence	The sealed intended takeaway, the reader's verbatim three-second takeaway, the reading order observed, and the revisions that followed.
AI-use record	The Appendix D record identifier for every exchange that touched the artifact, and what was rejected.
Remaining limitations	What the artifact still does not establish, in the analyst's own words.
Analyst of record	Name and date. The person who circulates the artifact owns every encoding in it.
Reviewer	Name and date, where a second reader ran the checklist.

Three fields in that table are the ones usually left blank and are the ones worth insisting on. **Completed checklist** is what converts the four counts from a claim into a reference; the counts and the completed form must reconcile, and reconciling them takes ten seconds and is the only check anyone can run on the record itself. **Accepted defects** is what distinguishes a decision from an oversight, and its value is entirely in the sentence: a legend retained because the audience is technical and the canvas is crowded is a judgment; a legend retained because nobody looked is not, and the record is the only place the difference is visible. **Remaining limitations** is the field a reader of the artifact would most like to have and is the one an analyst under deadline is most tempted to write as "none." An artifact with no limitations has not been examined.

E.8 A Worked Sign-Off

The example below runs the checklist on an artifact Chapter 12 has already diagnosed in full, so that the worked example discloses nothing that is not already in the chapter. The artifact is **the four views reproduced and repaired in Lab 12.2**, not the six-chart deck they were taken from. That distinction is the example's first lesson: a sign-off covers what was audited, and signing for six views on the strength of four is precisely the overreach check 2.15 and check 6.5 exist to catch. The two views Lab 12.2 does not reproduce are a separate artifact and need their own record before anything is claimed about them.

It is written as the record would look after Lab 12.2 Part B, with the repair notes carried across from Section E.6. Its counts are worth reading closely, because the short form for this artifact class is E.5.1 run four times, and the arithmetic below is what makes the sign-off checkable.

Table E.25

A completed sign-off record for the four views of Lab 12.2

Field	Entry
Artifact title	Suburban store performance — the four views reproduced and repaired in Lab 12.2
File and version	suburban_deck_v1, four Tableau worksheets, delivered as PNG exports. Two further views travelled in the same email and are not covered by this record
Data refresh date	Certified extract, line grain and order grain. No refresh date appeared on any of the four as delivered
Population and window	Suburban and resort store types. Two most recent quarters on view one; full certified window on views two through four. Neither the population nor the window was stated on any of the four
Instrument run	Baseline audit: the applicable master checkpoints run in full on each of the four views. Front-end form: E.5.1, run once per view
Completed checklist	appendix_e_signoff_suburban_deck_v1.xlsx, tabs V1 through V4
Checkpoints run	E.1, E.2, E.3, E.4, E.5, E.6, E.7, E.11, E.13, E.14. E.8 not applicable: separate views, no canvas. E.9 not applicable: static exports, no controls. E.10 not applicable: no forecast, estimate, or threshold. E.12 not applicable: working exchange between colleagues, no reader test required
Unique applicable check IDs	89, after the four not-applicable checkpoints and the per-view exclusions recorded on each tab. Counted once for the artifact; the 4 views produce 274 individual check instances, which live on tabs V1 through V4
Unique check IDs passed	71
Unique Level 2 check IDs accepted	1
Unique Level 1 check IDs repaired	17
Unique Level 1 check IDs unresolved	0
Stop-ship failures found	1.1 and 1.3 — no decision sentence, and the views' job never declared before they were forwarded as decision support. 2.1, 2.2, and 2.3 — source, window, and filters stated on none of the four. 2.4 and 2.6 — a per-order claim carried on line-grain marks, unverified by hand. 2.12 — no refresh date. 4.3 — the pie puts a magnitude the reader must rank onto angle. 5.1 — view one's axis runs from $240,000, drawing the second bar at roughly twelve times the first against a value ratio of 1.046 to one. 5.8 — view two's dual axis carries two independently scaled ranges. 6.10 — no footer on any of the four. 6.11 — the covering email calls view four "average order value" while the worksheet computes an average of line revenue. 7.1 and 7.2 — a red-to-green palette on an ordered field, with hue carrying the distinction alone. 14.2 and 14.6 — the delivered exports carry neither footers nor a refresh date. Seventeen distinct checks, failing across the four views on 31 instances in all; every instance repaired before recirculation, with its repair note on tabs V1 through V4
Communication revisions	3.9 — the pie replaced by a sorted bar and then by a 100 percent stacked bar split by metro. 3.6 — sort applied to the rebuilt bar. 7.8 — accent hue reserved for the series under discussion on the rebuilt trend view
Accepted defects	11.11 accepted as Caution: no list of charts built and rejected was kept, because the views were drafted in one pass and no alternatives were built. Recorded rather than repaired, and it is the reason the next exchange starts with a specification
Sentence count	Five claim-making sentences checked against 6.2 and 6.4 — four titles and the covering email. Two changed; the email's "average order value" sentence withdrawn
Removal list	Nothing removed. The pie was replaced rather than cut, and every repair changed an encoding rather than the evidence on display
Usability evidence	Not applicable; a working exchange between colleagues, and the reader test of Checkpoint E.12 is reserved for decision artifacts

Field	Entry
AI-use record	Appendix D record for the original request, the response, and the four repairs. Checkpoint E.13: 13.1 failed on the original exchange, since no faults were predicted before the specification was built, and passed on the graded re-run; 13.2 passed, the specification having been built as written with its faults intact; 13.5 failed on four inherited defaults; 13.12 and 13.13 completed
Remaining limitations	The repaired views establish that suburban and resort revenue rose 4.6 percent quarter over quarter. They do not establish that the movement is larger than ordinary quarter-to-quarter variation in this series, and the eight-quarter trend view built in the repair is the artifact that would answer it. Nothing here covers the two views that travelled in the same email
Analyst of record	[Name], [date]
Reviewer	[Name], [date]

Two things about that record are worth naming. The first is arithmetic: 71 passed plus 1 Caution plus 17 Stops repaired plus 0 unresolved is 89, which is the unique-applicable figure, and a reader who cannot make those five numbers reconcile has found a defect in the record rather than in the artifact. That is the whole purpose of the count line, and it is a check anyone can run in ten seconds without seeing the views. Note what the line counts and what it does not: 6.10 failed on all four views and appears once here as one identifier, with four instances and four repairs recorded on the tabs.

The second is why the record is kept at all. The review cost roughly thirty minutes against material that had already reached the director of real estate, and what it produced of lasting value is not the four repaired views — those will be superseded within a week — but the line in the limitations field, which says what the corrected evidence does and does not establish, and the Caution on 11.11, which says what the next exchange should start with. Neither sentence existed before the checklist was run.

E.9 Checkpoint Index by Chapter Section

Table E.26 is the reverse lookup. A student working through a chapter section can find the checkpoint that enforces it without reading this appendix from the front, and an instructor can confirm that a chapter's obligations are represented.

Table E.26

Chapter sections and the checkpoints that enforce them

Chapter section	What it establishes	Checkpoint
2.7; 12.9	The baseline discipline, and its visual form as the reference line	E.5
3.3; 3.4; 12.10	Grain, levels of measurement, and aggregation in the view	E.2, E.7
12.1; 12.2	Visualization's two jobs, and the drift from the first to the second	E.1
12.3; 12.4	Marks, encodings, and the perceptual hierarchy	E.4
12.5	Chart selection by analytic purpose	E.3
12.6; 12.7	Scales, the zero-baseline obligation by encoding, and the five mechanics	E.5
12.8	Color as an encoding system, and accessibility as correctness	E.7
12.9	Reference lines, annotation, and the footer discipline	E.5, E.6, E.14
12.11	Faceting and small multiples	E.3, E.4
12.12	The AI chart-drafting routine and its nine failure modes	E.13
12.16	The disclosure standard and the duty to inspect	E.11
13.2	Audience analysis and the three-second test	E.1, E.12
13.3	Exploratory and explanatory artifacts, and choosing the form	E.1

Chapter section	What it establishes	Checkpoint
13.4; 13.5	The KPI hierarchy and its expression in layout	E.8
13.6	The assertion title	E.6
13.6	Narrative order across a set of views	E.8
13.7	Preattentive emphasis and data-ink decluttering	E.7, E.8
13.8	Interactivity that communicates and interactivity that conceals	E.9
13.5; 13.12.4	Dashboard sizing, containers, the title band, and alternative text	E.8, E.14
13.9	Uncertainty in decision units, and the four-part recommendation	E.10
13.10	The memo, the usability test, and the critique protocol	E.12
13.11	The AI communication routine and its eight failure modes	E.13
13.15	Emphasis, distortion, and the counterfactual test	E.11
Table 8.5 (three transferable points)	Frame, baseline, and error in decision units	E.1, E.5, E.10
Table 12.5 (V1–V3)	Grain and aggregation; scale and baseline honesty; encoding fits the purpose	E.2, E.5, E.3 and E.4
Table 13.6 (C1–C3)	Every sentence earned; the reader's takeaway; what was removed is disclosed	E.6, E.12, E.11

E.10 Exclusion Register

A checklist that silently omits an obligation looks identical to one that never considered it. Table E.27 records what this appendix does not contain and why, because several of the entries are decisions rather than oversights.

Table E.27

What this appendix does not contain, and why

Not included	Reason
Chart-selection theory, the perceptual hierarchy's evidence, and the grammar of a chart	Chapter 12 owns all three. Checkpoints E.3 and E.4 enforce the conclusions and cite the sections; an appendix that re-argued them would be a second chapter and would drift from the first.
Upstream data-quality checks	Chapter 4 owns the verification log and the defect catalog. Checkpoint E.2 checks that the artifact is consistent with a certified source; it does not certify the source.
Metric definitions	Appendix B is the canonical dictionary. Checkpoint E.6 checks that a definition accompanies a number a reader could misread; it does not supply the definition.
Statistical guidance on choosing an interval	Chapters 10 and 11 own the intervals. Checkpoint E.10 checks that the interval that exists is displayed honestly and converted correctly.
Prompts for AI assistance	Appendix C owns the templates. Checkpoint E.13 is the audit that runs on their output, and Templates C.16 through C.18 route back here for the repair.
A pass/fail score, a weighting, or a grade	The checklist distinguishes hard errors from judgment calls and records both. Converting that into a single number would suppress the distinction, which is the appendix's most useful property.
Project rubrics	The rubrics are a separate instrument and are released with the project briefs. This appendix is incorporated into Project #2's submission workflow as the sign-off record of Section E.7, whatever the rubrics' final home.
Screenshots of software interfaces	Numbered checks and build paths in the chapters are more durable and more accessible than an image of a menu that will be redrawn.

Not included	Reason
Vendor-specific rules	Every check is written to be tool-neutral. Where a Tableau behavior is named — reference bands, filter scope, alt text, action matching — it is because the behavior is what makes the check necessary, and the vendor's documentation governs the current interface.

One dependency is recorded rather than excluded. Checkpoint E.13 requires an Appendix D record for every AI-assisted exchange, and Chapters 12 and 13 both make the same requirement at the point where the exchange occurs. Appendix D holds the canonical field set — the exchange identifier, the stage, the tool and version, the prompt as sent, the output received, the disposition, the error or limitation found, the verification performed, the change to the final work, and the analyst's own judgment — and Checkpoint E.13 uses those fields rather than defining its own. Where an artifact's AI record and its visual sign-off are submitted together, the exchange identifier is what links a repair note in Section E.6 to the exchange that made it necessary.

References

Cleveland, W. S., & McGill, R. (1984). Graphical perception: Theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association, 79*(387), 531–554. https://doi.org/10.1080/01621459.1984.10478080

Correll, M., Bertini, E., & Franconeri, S. (2020). Truncating the y-axis: Threat or menace? In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376222

Correll, M., & Gleicher, M. (2014). Error bars considered harmful: Exploring alternate encodings for mean and error. *IEEE Transactions on Visualization and Computer Graphics, 20*(12), 2142–2151. https://doi.org/10.1109/TVCG.2014.2346298

Doncaster, P. (2014). *The UX five-second rules: Guidelines for user experience design's simplest testing technique*. Morgan Kaufmann.

Few, S. (2006). *Information dashboard design: The effective visual communication of data*. O'Reilly Media.

Few, S. (2012). *Show me the numbers: Designing tables and graphs to enlighten* (2nd ed.). Analytics Press.

Healey, C. G., & Enns, J. T. (2012). Attention and visual memory in visualization and computer graphics. *IEEE Transactions on Visualization and Computer Graphics, 18*(7), 1170–1188. https://doi.org/10.1109/TVCG.2011.127

Heer, J., & Bostock, M. (2010). Crowdsourcing graphical perception: Using Mechanical Turk to assess visualization design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 203–212). Association for Computing Machinery. https://doi.org/10.1145/1753326.1753357

Heer, J., & Shneiderman, B. (2012). Interactive dynamics for visual analysis. *Communications of the ACM, 55*(4), 45–54. https://doi.org/10.1145/2133806.2133821

Hullman, J., & Diakopoulos, N. (2011). Visualization rhetoric: Framing effects in narrative visualization. *IEEE Transactions on Visualization and Computer Graphics, 17*(12), 2231–2240. https://doi.org/10.1109/TVCG.2011.255

Knaflic, C. N. (2015). *Storytelling with data: A data visualization guide for business professionals*. Wiley.

Krug, S. (2014). *Don't make me think, revisited: A common sense approach to web usability* (3rd ed.). New Riders.

Munzner, T. (2014). *Visualization analysis and design*. CRC Press.

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 206–213). Association for Computing Machinery. https://doi.org/10.1145/169059.169166

Pandey, A. V., Rall, K., Satterthwaite, M. L., Nov, O., & Bertini, E. (2015). How deceptive are deceptive visualizations? An empirical analysis of common distortion techniques. In *Proceedings of the 33rd Annual*

ACM Conference on Human Factors in Computing Systems (pp. 1469–1478). Association for Computing Machinery. https://doi.org/10.1145/2702123.2702608

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages* (pp. 336–343). IEEE. https://doi.org/10.1109/VL.1996.545307

Spiegelhalter, D., Pearson, M., & Short, I. (2011). Visualizing uncertainty about the future. *Science, 333*(6048), 1393–1400. https://doi.org/10.1126/science.1191181

Tableau. (2026a). *Actions*. Tableau Help. Retrieved August 2, 2026, from https://help.tableau.com/current/pro/desktop/en-us/actions.htm

Tableau. (2026b). *Author views for accessibility*. Tableau Help. Retrieved August 2, 2026, from https://help.tableau.com/current/pro/desktop/en-us/accessibility_create_view.htm

Tableau. (2026c). *Filter data from your views*. Tableau Help. Retrieved August 2, 2026, from https://help.tableau.com/current/pro/desktop/en-us/filtering.htm

Tableau. (2026d). *Reference lines, bands, distributions, and boxes*. Tableau Help. Retrieved August 2, 2026, from https://help.tableau.com/current/pro/desktop/en-us/reference_lines.htm

Tableau. (2026e). *Size and lay out your dashboard*. Tableau Help. Retrieved August 2, 2026, from https://help.tableau.com/current/pro/desktop/en-us/dashboards_organize_floatingandtiled.htm

Treisman, A. M., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12*(1), 97–136. https://doi.org/10.1016/0010-0285(80)90005-5

Tufte, E. R. (2001). *The visual display of quantitative information* (2nd ed.). Graphics Press.

Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science, 211*(4481), 453–458. https://doi.org/10.1126/science.7455683

W3C. (2024). *Web content accessibility guidelines (WCAG) 2.2* (W3C Recommendation, December 12, 2024). World Wide Web Consortium. https://www.w3.org/TR/WCAG22/

Wong, B. (2011). Points of view: Color blindness. *Nature Methods, 8*(6), 441. https://doi.org/10.1038/nmeth.1618